Recherche de motifs dans des images : apport des graphes plans

Émilie Samuel

Laboratoire Hubert Curien, UMR CNRS 5516
Faculté des Sciences et Techniques
École doctorale 488 Sciences, Ingénierie, Santé

Thèse présentée
pour obtenir le grade de :

Docteur de l'Université Jean Monnet de Saint-Étienne
Mention Informatique

Recherche de motifs dans des images : apport des graphes plans

Émilie Samuel

Soutenue le 06 juin 2011 devant le jury composé de :

Michel Habib	Paris Diderot	Président
Luc Brun	Caen	Rapporteurs
Lhouari Nourine	Clermont-Ferrand	
Christine Largeron	Saint-Étienne	Examinateurs
Christian Wolf	Lyon	
Colin de la Higuera	Nantes	Directeur
Jean-Christophe Janodet	Saint-Étienne	Co-directeur

Remerciements

Il est temps pour moi de conclure ces années de thèse, et de remercier comme il se doit toutes les personnes ayant contribué, de façon plus ou moins directe, à la réalisation de ces travaux. Je veux avant tout exprimer ma gratitude à tous ceux qui, par leur présence, m'ont aidée. Que l'on m'excuse par avance des oublis que je risque de commettre...

Je voudrais commencer par témoigner ma reconnaissance envers Colin de la Higuera, mon directeur de thèse, et Jean-Christophe Janodet, mon co-directeur.

Colin pour avoir toujours été disponible, pour son engagement et son exigence, et pour m'avoir accordé la liberté et l'autonomie dont j'avais besoin.

Jean-Christophe pour son entrain et son dynamisme sans pareils, ses conseils (pas toujours suivis à la lettre ;)) et son soutien, surtout cette dernière année.

Merci à tous les deux de m'avoir guidée avec patience, et d'avoir été compréhensifs vis-à-vis de mes choix personnels, qui n'étaient pas toujours en accord avec les vôtres. Les travaux présentés dans cette thèse vous doivent évidemment beaucoup.

J'aimerais ensuite adresser mes remerciements aux membres du jury, pour avoir accepté d'évaluer mon travail, et pour leur implication. Merci à Michel Habib d'avoir présidé ce jury, et aux rapporteurs, Luc Brun et Lhouari Nourine, pour les remarques pertinentes et constructives faites sur mon mémoire et pour le temps qu'ils y ont consacré. Merci également aux examinateurs, Christine Largeron et Christian Wolf, pour leurs commentaires.

Au-delà de la soutenance, la thèse fut une drôle d'aventure, partagée du début à la fin avec deux compagnons de route, Christophe et Laurent. Je crois que nous avons souvent fait face à des doutes et interrogations similaires, et pouvoir les partager a constitué pour moi un véritable soutien. Merci pour la complicité et pour avoir partagé mon bureau presque en continu, avec tout ce que cela implique :p. Notez bien que je suis toujours partante pour se partager quelques millions un prochain samedi.

J'en arrive à tous les membres de l'équipe, que ce soient ceux qui étaient là avant et le seront encore après, ceux qui sont venus puis repartis, ou encore la relève. Un merci particulier à ceux qui ont partagé mon bureau et mes repas, avec qui j'ai discuté et travaillé. Merci, donc, à : Adriana, Aurélien, Baptiste, Catherine, Chahrazed, Christine, David, Élisa (Miss Annie te salue), Fabien, Fabrice A., Fabrice M., Florian, François, Franck, Frédéric (et ses innombrables liens qui ne sont jamais tous les *meme*, ses réponses de sage et sa bible de raccourcis *emacs*, pour ne citer que cela), Hazaël, Jean, Jean-Philippe, Julien, Léo, Marc B. (pour les discussions, et cette autre vision de la recherche), Marc S., Mathias, Mattias, Pierre, Philippe, Richard, Sabri, Stéphanie, Thierry (pour, entre autres, *speedy*, *jerry*, et surtout *mooch*, qui m'accueille tous les matins), Tung, et

les autres...

Je n'oublie pas le personnel adminsitratif du laboratoire : Amélie, Anne-Laure, Claude, Dalila, Éric, Jeanine, Patricia C., Patricia L., Patrick, Philippe, et Stéphanie.

Je voudrais également remercier toutes les personnes qui ont été impliquées dans le projet ANR SATTIC, avec qui j'ai travaillé de près ou de plus loin, et que je n'ai pas encore citées : Camille, Cécile, Christine, Christophe, Guillaume, Jean-Michel, Marc, Sébastien, et Stéphane.

Je terminerai ces remerciements par mes proches, tous ceux qui, non des moindres, ont été à mes côtés pendant ces années, ou l'auraient été s'ils l'avaient pu. Merci de m'avoir soutenue, comprise et encouragée. Merci à mes parents d'avoir supporté mes sautes d'humeur. Merci à ceux avec qui j'ai partagé mes interrogations existentielles, et qui m'ont fait me rappeler qu'il y avait tellement d'autres choses à voir et prendre en considération.

Et, pour conclure à la manière de Binet :

Cette thèse est dédicacée :

Aux pauses de 10h26, pauses-croissants, pauses-gâteaux, pauses-pulco, pauses-jeux, et autres pauses-pauses.

Aux auteurs de chevet m'ayant accompagnée et fait respirer pendant ces années.

Cette thèse n'est pas dédicacée :

À la disparition, par une sombre nuit d'hiver, des décorations de l'arbre de Noël.

Au publish or perish.

D'une façon générale, cette thèse est dédicacée à toutes les personnes avec qui mes rapports furent aussi divers qu'enrichissants.

Table des matières

Introduction

Ceci n'est pas un manchot

René Magritte l'aurait peut-être dit : « Ceci n'est pas un manchot »[1].

Pourtant, au premier regard, chacun d'entre nous a sans doute conjecturé, sans plus de réflexion, qu'il s'agissait d'un manchot, ou peut-être d'un pingouin. Mais peu se seront instantanément dit « Ceci n'est pas un manchot, ceci est l'image d'un manchot, sa représentation ». Car assurément, ceci n'est pas un manchot. Tout comme la pipe de Magritte qui ne pouvait être ni bourrée ni fumée, ce manchot ne peut se nourrir, nous ne pouvons l'entendre crier.

Cette image, qui se trouve être une photographie, n'est qu'un ensemble d'éléments graphiques évoquant le réel. Et c'est l'observateur qui l'analyse et y assigne un sens, comme l'expliquait Umberto Eco [Eco, 1970] en parlant d'une annonce publicitaire : « Mais en réalité, quand je vois un verre de bière (vieille question psychologique qui emplit l'histoire de la philosophie) je perçois bière, verre et fraîcheur, mais je ne les sens pas : je sens au contraire quelques stimuli visuels, couleurs, rapports spatiaux, incidences de lumière, *etc.* (donc déjà coordonnés dans un certain champ perceptif), et je les coordonne (dans une opération transitive complexe) jusqu'à ce que s'engendre une structure perçue qui, sur la base d'expériences acquises, provoque une série de synesthésies et me permet de penser : « bière glacée dans un verre ». »

Les représentations des choses deviennent les choses : l'homme interprète les éléments qu'il reçoit, et leur assigne une signification en les mettant en correspondance avec ses connaissances. Par notre expérience commune, nous pouvons donner un sens à des éléments figuratifs : Eco parle de *codes de reconnaissance*. Quotidiennement, nous percevons, sans avoir conscience de cette mécanique sous-jacente.

[1] Référence au tableau *La Trahison des images* de René Magritte (1929, huile sur toile, 59 x 65 cm, Art Institute of Chicago), légendé "Ceci n'est pas une pipe".

Confronté à cette image, l'ordinateur, lui, serait sans doute plus proche de la vision de Magritte. Il ne nous répondrait bien sûr pas que ceci n'est pas un manchot, mais encore moins que ceci est un manchot. Par contre, il en resterait à ce qu'il sait : ceci est une image, c'est-à-dire un ensemble de pixels, ayant chacun des caractéristiques et formant un tout. Ce que le tout représente, il ne le sait pas.

Pour l'homme, la perception des images va encore plus loin : confrontés à un ensemble d'images, nous avons la capacité de les classer en catégories. Cette classification peut, par exemple, être déduite du genre des images : s'agit-il de photographies, de peintures, de schémas ? Mais le classement est souvent bien plus subtil, liant signification et hiérarchie. Dans la catégorie mammifères, nous délimiterons les oiseaux, avant de nous restreindre à ceux qui ne peuvent voler, pour aboutir (probablement après d'autres subdivisions) aux manchots. Sans compter que nous pourrons également faire intervenir des notions de taille, couleur, texture, orientation, forme, échelle...

S'il est impossible pour un ordinateur de savoir, sans information extérieure, que l'image représente un manchot, on peut par contre espérer le lui faire déduire, et ce de la même façon que nous : en utilisant des connaissances antérieures. Ainsi, si nous lui fournissons un ensemble d'exemples, lui apprenant que telles images représentent des manchots, telles autres des arbres, et telles autres des bicyclettes, alors, nous pouvons ensuite lui donner pour règle que « tout ce qui ressemble plus à un manchot qu'au reste est un manchot » (les exemples d'apprentissage devant couvrir le champ des possibilités). Reste pour cela à lui donner les moyens de pouvoir quantifier cette ressemblance.

Bien entendu, la tâche est plus complexe que cela peut paraître. Tout d'abord, nous l'avons dit, pour un ordinateur, toute image est un ensemble de pixels. Pour lui, donc, pas de différence entre le dessin et la photographie ; ni entre le manchot et la bicyclette. D'autant plus que graphiquement, il existe une infinité de moyens pour représenter un manchot, le suggérer, le symboliser. Et il existe une infinité de manchots possibles : manchot de profil, debout, nageant, la tête baissée, adulte ou jeune... Pour répondre à ces difficultés, deux sortes de méthodes existent : la première vise à travailler directement sur la représentation par pixels et à l'exploiter au mieux. La seconde se propose plutôt de définir des représentations alternatives des images, qui intègrent la sémantique. C'est cette seconde voie que nous emprunterons dans ce manuscrit.

En effet, les pixels sont une forme de représentation obtenue par un dispositif physique (par exemple, un appareil photo). Mais ce n'est qu'une possibilité de représentation parmi d'autres, qui *a priori* n'a pas à être privilégiée pour exploiter une image. Tout autre forme de représentation est légitime sous réserve qu'elle soit intéressante et puisse apporter des informations complémentaires. Le but final n'est donc pas d'opposer les matrices de pixels à d'autres modèles symboliques, mais bien, dans l'idéal, d'être capable d'allier les deux pour tirer parti de leur complémentarité.

Le nouveau modèle de représentation se doit de pouvoir être extrait de toute image, tout en évitant d'être trop complexe : sa création, son stockage, son analyse doivent pouvoir être réalisés en temps raisonnable. Mais, ce critère de simplification n'est pas réellement une contrainte en soi. En effet, comme l'explique Eco, l'homme n'a pas besoin de tous les détails pour assigner un sens : « [...] nous avons dit aussi qu'un signe peut dénoter globalement un perçu, réduit à une convention graphique simplifiée. C'est

précisément parce que, parmi les conditions de la perception, nous choisissons les traits pertinents, que ce phénomène de réduction se vérifie dans la quasi-totalité des signes iconiques [...]. » L'image est information, cependant une partie d'entre elle est superflue dans le sens où elle n'est pas nécessaire à sa compréhension, et peut être perdue ou ignorée au profit de la mise en exergue des informations pertinentes et nécessaires.

Puis vient la mise en correspondance des modèles : puisqu'ils renferment l'information fondamentale des images qu'ils représentent, c'est désormais directement sur eux qu'il faut concentrer les efforts. Ceci passe par l'élaboration d'une procédure appropriée et efficace permettant de vérifier d'une part si des modèles sont les mêmes (ce qui signifie que les images sous-jacentes ont une sémantique proche), s'ils sont inclus les uns dans les autres (le concept représenté sur telle image fait partie de ceux contenus dans telle autre image) ou s'ils se ressemblent (les images dont ils sont issus ont une sémantique proche).

Un mode de représentation intégrant la sémantique

Le travail présenté dans ce manuscrit va s'attacher à s'abstraire de la représentation d'images basée sur des pixels, et à se donner un autre modèle de représentation. Celui-ci devra, dans la mesure du possible, être dépourvu d'information superflue, tout en conservant celle qui est essentielle. Il sera en outre nécessaire de s'assurer que ce modèle puisse être extrait de façon universelle, quelque soit l'image considérée. Enfin, il nous faudra trouver une solution efficace pour pouvoir associer entre eux les modèles – ou parties de modèles – et mettre en avant leur similarité.

Parmi les représentations alternatives possibles, nous baserons notre travail sur une théorie informatique et mathématique connue, la théorie des graphes. Les graphes sont des modèles de représentation souvent utilisés. Ils peuvent ainsi figurer des réseaux (informatiques, électriques, sociaux, économiques, de transport... [Milgram, 1967, Hong et al., 2004]), être utilisés en logistique (planification, stockage, routage), en biologie et en chimie [Bonchev et Rouvray, 1991], en cartographie [Dent et al., 2008], en architecture [Franz et al., 2005], en linguistique [Nivre et McDonald, 2008], en bases de données, mais aussi pour modéliser des problèmes (problème des quatre couleurs [Robertson et al., 1997], du voyageur de commerce [Applegate et al., 2006]...). L'attention s'est portée depuis plusieurs années sur la modélisation d'images sous forme de graphes et sur leur utilisation pour la reconnaissance de formes [Conte et al., 2007]. En outre, on assiste à la création de bases de données de graphes, soit générés artificiellement [De Santo et al., 2003], soit issus de données réelles [Riesen et Bunke, 2008] (lettres manuscrites, empreintes digitales, symboles, objets, document *Web*, molécules et protéines) dédiées au développement d'algorithmes.

Les graphes peuvent être complexes. Mais nous l'avons dit, le modèle choisi ne doit pas l'être *trop*, ce pour des raisons d'efficacité. En outre, utiliser un modèle trop riche risque d'avoir un autre désavantage : masquer le graphe lui-même, et utiliser donc davantage les données ajoutées que le graphe.

Nous considérerons donc des graphes constitués de deux entités : les sommets et les

arêtes, dénotant respectivement des éléments, et les relations entre eux. De tels graphes ont été – et sont – l'objet d'études algorithmiques notoires (recherche de plus court chemin, coloration, détermination de flux maximal, parcours en profondeur et largeur, fermeture transitive... [Cormen et al., 1990]). Les graphes que nous étudierons auront une particularité supplémentaire, celle de la planarité. Plus précisément, il s'agira de graphes plans, c'est-à-dire de graphes plongés dans le plan, de façon à ce qu'aucune de leurs arêtes ne se croise. Nous le verrons, cette particularité est en fait assez naturelle lorsqu'il s'agit de représenter des images.

La reconnaissance de formes est un des domaines d'application principaux de la représentation d'images sous forme de graphes. Elle vise à identifier des motifs présents dans des images, de façon à pouvoir les mettre en rapport les unes avec les autres. Il s'agit donc de considérer tout graphe comme étant un ensemble de motifs, et de chercher à retrouver tout ou partie de ces motifs (rigoureusement identiques ou légèrement modifiés) dans d'autres graphes. Cette tâche est intimement liée à la classification d'images, qui vise à constituer des classes d'images selon des critères prédéfinis (on pourrait donc arriver à créer automatiquement la classe des manchots), ou encore à assigner de nouvelles images à des classes existantes (tout ce qui serait plus proche de la classe des manchots que des autres classes serait un manchot). Ces applications peuvent être considérées comme des sous-domaines de l'apprentissage automatique, où l'ordinateur apprend à généraliser à partir des connaissances et règles fournies et déduites de son expérience. De nombreuses applications de tels travaux existent. Il peut par exemple s'agir de recherche d'information (nous le verrons dans la partie suivante pour les moteurs de recherche), d'analyse d'images médicales afin de détecter des anomalies (diagnostic assisté), de contrôles qualité industriels, de détection automatique de contrefaçons (par recherche de motifs), ou encore de reconnaissance de visages ou d'attitudes.

Une autre étape importante est celle de la mise en relation des modèles et de la découverte de caractéristiques communes. Pour deux graphes donnés, il peut s'agir de mettre en correspondance les sommets du premier graphe avec ceux du second. Une telle association dépend à la fois des attributs des sommets, et des arêtes pouvant exister. Elle peut être envisagée comme un alignement des sommets (semblable à l'alignement de séquences d'ADN pour la découverte de ressemblances structurelles en bioinformatique, voir la figure 0.i) contrainte par le respect des arêtes existant entre eux.

FIG. 0.i – Alignement de trois séquences d'ADN : certaines bases sont en trop, ou manquantes, ou différentes.

Cette mise en correspondance des sommets en accord avec les arêtes qui les relient est appelée appariement de graphes *(graph matching)*. Il est évident que pour deux graphes, de nombreux appariements de sommets sont possibles. Parmi eux, certains

sont meilleurs que d'autres, parce qu'ils respectent mieux les attributs des sommets, ou les arêtes. Dans l'idéal, c'est le meilleur appariement que l'on souhaite trouver, afin de dire avec précision si les graphes – et donc les images – se ressemblent. Étudier les critères entrant en compte dans le calcul de la qualité d'un appariement sera une des tâches décrites dans cette thèse.

En fait, l'appariement de graphes est un terme qui englobe plusieurs problématiques. Parmi elles, nous concentrerons nos efforts en particulier sur le développement d'algorithmes d'isomorphisme de graphes et de sous-graphes, permettant de conclure sur le fait que deux graphes puissent être les mêmes (les images ont la même sémantique), ou que l'un puisse être inclus à l'intérieur de l'autre (le concept d'une image est un des concepts d'une autre image).

Intérêt et utilité d'un mode de représentation intégrant la sémantique des images

Pourquoi veut-on amener un ordinateur à être capable de raisonner sur la sémantique des images ? Car elles nous entourent, et, constituant plus qu'un détail, elles prennent peu à peu le pas sur le texte, bien qu'ils restent complémentaires[2]. Utilisées pour la pédagogie, elles visent à transmettre des connaissances, expliquer, illustrer, compléter. Les images journalistiques tendent elles à informer, décrire des faits, transmettre des données en représentant le réel (ce qui n'exclut pas des choix de composition ou d'instants). Les publicitaires les utilisent pour attirer le regard, mettre en situation, argumenter et convaincre...

Or, toutes ces images nous sont présentées à la fois de façon tangible, réelle (dans des magazines, des prospectus, des affiches, des catalogues, des livres...) mais aussi virtuelle, et c'est sans doute à ce niveau qu'elles sont de plus en plus présentes. Les ordinateurs personnels, mais surtout Internet, ouvrent des possibilités et véhiculent un nombre croissant d'images numériques. Et ces dernières ne sont plus un simple complément : la recherche d'information, qui auparavant ne se basait que sur les données textuelles, les intègre aujourd'hui [Barnard et al., 2003, Moulin et al., 2010]. D'autre part, nous le savons, certains moteurs de recherche consacrent une partie de leurs services à la recherche d'images. Le très usité Google[3] propose ainsi de retrouver des images à partir de mots-clés (voir la figure 0.ii), puis de les filtrer par taille, couleur, mais aussi de ne sélectionner que des visages, des photographies, ou des dessins.

Mais certains moteurs vont encore plus loin, comme TinEye [4] qui renvoie, à partir d'une image ou de son URL, la même image et/ou des versions modifiées (meilleure définition, changement de cadrage, de couleur, obstruction...) ; ou encore GazoPa [5] qui renvoie des images similaires (selon des critères tels que la couleur ou la forme) et propose

[2] Claude Cossette, Les images fonctionnelles http://www.comviz.com.ulaval.ca/module1/1.4_fonction.php

[3] http://www.google.com

[4] http://www.tineye.com/

[5] http://www.gazopa.com/

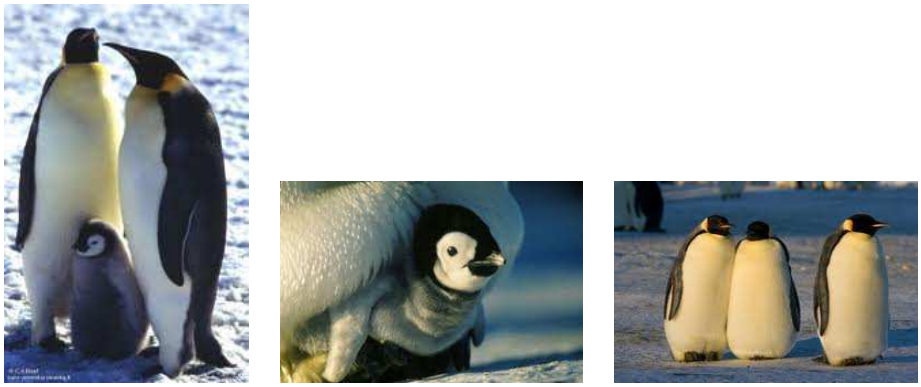


FIG. 0.ii – Premiers résultats de Google à la requête « manchot » (aucun critère particulier spécifié).

même aux utilisateurs de dessiner leur requête. L'image n'est donc plus seulement le résultat d'une recherche, mais également sa donnée (voir la figure 0.iii).

FIG. 0.iii – Premiers résultats de TinEye (1ère ligne) et de Gazopa (2nde ligne) à la requête correspondant à l'image du manchot.

Les résultats de ces outils[6] sont intéressants mais restent limités. TinEye ne permet que très peu de généralisation puisqu'il renvoie exactement la même image modifiée : aucune possibilité, donc, de trouver l'image d'un autre manchot. Quant à GazoPa, les critères utilisés n'intègrent visiblement pas – ou peu – de sémantique : les résultats sont cohérents vis-à-vis des couleurs, mais pas des objets représentés. L'intérêt du développement d'une méthode unissant les points forts de ces approches est certain.

[6]Ces images sont présentées à titre d'illustration seulement. Les résultats varient en fonction de la date de la recherche et de ses paramètres.

Contexte de l'étude

Le sujet de cette étude est au cœur des préoccupations du projet ANR blanc SATTIC[7], mené en partenariat avec le Laboratoire Hubert Curien de Saint-Étienne et le LIRIS de Lyon. Le postulat de départ est qu'il est nécessaire de développer un système de représentation riche permettant de décrire les images, tout en étant mathématiquement bien défini et exploitable informatiquement de façon efficace. Initialement, nous nous sommes essentiellement focalisés sur les images de type vignette, de faible définition (comme celles renvoyées par une recherche sur Google Images, par exemple) et sur l'utilisation de points d'intérêt. En effet, même si cette question dépasse le cadre de ce manuscrit, il semble que dans ce contexte, les méthodes numériques utilisées en image soient pénalisées. Les méthodes symboliques peuvent constituer alors une alternative à étudier.

Plusieurs étapes ont ainsi été définies :

- choix de structures symboliques et définitions de mesures permettant d'évaluer les distances entre images ;
- caractérisation statistique de ces structures symboliques ;
- définition d'algorithmes sur ces structures permettant le calcul de mesures de distances.

Le travail présenté dans ce manuscrit a principalement œuvré pour atteindre les premier et troisième buts.

Organisation du manuscrit

Ce manuscrit est organisé en cinq chapitres.

Le premier est consacré à la représentation d'images sous diverses formes constituant des alternatives aux matrices de pixels. Pour cela, nous évoquerons les principales notions sur les images numériques qui seront utiles à notre étude. Nous aborderons notamment la segmentation et l'extraction de caractéristiques. Puis nous nous focaliserons sur les représentations structurées d'images, des histogrammes aux chaînes, et aux arbres, pour aboutir aux graphes.

Le deuxième chapitre sera centré sur les appariements de graphes, et plus précisément sur l'étude de critères permettant d'élaborer une fonction de coût d'appariements. Nous verrons que cette approche, devant permettre de discriminer un mauvais appariement d'un meilleur, et donc d'entrer en possession de moyens permettant d'optimiser les appariements de graphes, n'a pas abouti. Nous réorienterons nos travaux dans les chapitres suivants, pour nous recentrer sur la structure des graphes.

Nous montrerons dans le troisième chapitre ce que la théorie des graphes peut apporter au domaine de l'image. Nous introduirons la terminologie nécessaire sur les graphes et la planarité. Puis nous aborderons plus particulièrement l'isomorphisme de graphes et de sous-graphes. Nous expliquerons comment les utiliser dans le contexte de la reconnaissance de formes. Enfin, nous évoquerons également les problèmes de plus grand

[7] http://labh-curien.univ-st-etienne.fr/wiki-sattic/index.php/Main_Page

sous-graphe commun et de distance d'édition de graphes, témoins de la similarité entre images.

Le chapitre quatre sera consacré à la présentation d'une nouvelle méthode d'extraction de graphes plans à partir d'images. Nous y exposerons notre cahier des charges, et ce qui fait l'originalité et l'intérêt de la méthode. Nous aborderons ensuite chacune des étapes menant à l'élaboration d'un graphe plan, avant de présenter une série d'expérimentations. Les travaux de ce chapitre ont donné lieu à une publication [Samuel et al., 2010].

Enfin, le dernier chapitre concernera l'élaboration d'algorithmes polynomiaux permettant de répondre aux problématiques d'isomorphisme de graphes et de sous-graphes particuliers : les graphes plans à trous. Nous définirons ces graphes, et nous intéresserons aux différents isomorphismes les concernant. Puis nous introduirons la notion d'équivalence de graphes plans à trous, et mettrons en évidence ses liens avec l'isomorphisme. Nous présenterons ensuite des algorithmes d'isomorphisme d'une part, et de recherche de motifs d'autre part, avant de proposer quelques expérimentations préliminaires dans le domaine de l'image. Les travaux de ce chapitre ont été publiés dans deux conférences [Damiand et al., 2009b, Damiand et al., 2009a], et un journal [Damiand et al., 2011].

1

Représentation d'images sous forme structurée

Sommaire

Avec les appareils photo numériques, les scanners, les logiciels de retouche d'images, Internet et les ordinateurs en général, il est devenu fréquent, voire quotidien, de manipuler ce que l'on appelle des images numériques. Dans ce but, elles sont stockées sous la forme d'ensembles de pixels. Cependant, si aujourd'hui ces images nous entourent, il est devenu nécessaire, pour les étudier plus attentivement, d'envisager des modèles de représentation alternatifs. En effet, des domaines de recherche tels la reconnaissance de formes ou la classification d'images requièrent une modélisation plus riche, plus structurée, pour donner automatiquement du sens à une image.

Dans ce premier chapitre, nous exposerons tout d'abord les notions de base sur les images numériques. Ainsi, après en avoir donné les principales caractéristiques, nous nous intéresserons à deux opérations de traitement d'images : la segmentation et l'extraction de caractéristiques. Par la suite, nous porterons notre attention sur plusieurs modèles symboliques de représentation des images numériques, permettant de structurer l'information qu'elles contiennent. Il s'agira des histogrammes, des chaînes, et des arbres. Nous terminerons notre étude par la représentation sous forme de graphes, qui sera le modèle privilégié de cette thèse.

1.1 Préambule sur les images numériques

Dans cette première partie, nous nous proposons de développer certaines notions de base sur les images numériques. Il s'agit de concepts, parfois communs pour un lecteur

familier du domaine, qui nous permettront d'appréhender au mieux les travaux exposés dans cette thèse. Nous nous intéresserons tout d'abord aux principales définitions en la matière, avant de détailler deux opérations usuelles de traitement d'images.

1.1.1 Définitions

Ce qu'est une image numérique

Une image numérique désigne une image acquise (par exemple par un appareil photo ou un scanner...), ou bien créée (notamment par un programme informatique ou une tablette graphique...), qui est ensuite stockée sur un support (comme un disque dur ou une clé USB...) et qui peut, par la suite, subir divers traitements (ce peut être une modification de taille, de couleur, ou bien d'autres choses encore). Cette image représente une scène qui, pour les besoins pré-cités de stockage, de représentation sur ordinateur, de manipulation ou d'analyse, se trouve décomposée en un ensemble de cellules dénombrables. On peut ainsi faire un parallèle entre cette façon de représenter des images et les mosaïques ou les vitraux, comparaison notamment proposée dans [Coeurjolly et al., 2007] (voir figure 1.i pour une illustration[1]). Cependant, ce rapprochement a des limites : dans le cas des mosaïques et vitraux, les différentes pièces composant la scène ont une forme qui s'ajuste à celle des objets représentés. Chaque pièce peut donc avoir sa propre forme pour mieux respecter les contours des objets.

FIG. 1.i – Découpe d'une mosaïque et d'un vitrail : exemple de scènes décomposées en un nombre fini d'éléments.

Le découpage n'est pas aussi libre pour les images numériques : elles sont composées d'un nombre fini d'éléments de même dimension formant la plus petite unité d'une image, appelés pixels (abréviation de la locution anglaise *picture element*, attribuée à F. C. Billingsley [Billingsley, 1965]), ayant chacun une position spatiale et une valeur. Cette décomposition est visible sur la figure 1.ii[2].

[1] Ces deux images sont sous licence Creative Commons, libres de reproduction, distribution et modification, et accessibles aux URLs suivantes : `http://www.flickr.com/photos/jfgornet/4124350882/` et `http://www.flickr.com/photos/dalbera/2131300627/`.

[2] Cette image est sous licence Creative Commons, libre de reproduction, distribution et modification,

FIG. 1.ii – Une image numérique : zoom sur la décomposition en pixels.

On peut donc, de façon plus formelle, considérer une image numérique comme étant une matrice de pixels. Dans la suite de cette thèse, nous utiliserons le terme image numérique pour désigner toute image vérifiant cette décomposition (par opposition, par exemple, aux images vectorielles qui sont décrites par des formules géométriques modélisant le tracé, et peuvent s'afficher sans perte à différentes échelles). De plus, nous nous intéresserons uniquement aux images en deux dimensions.

Notions associées

De nombreuses notions sur les images matricielles ont été introduites. Considérons tout d'abord les concepts de voisinage et d'adjacence de pixels [Coeurjolly et al., 2007]. Un pixel p de coordonnées (x, y) a quatre voisins horizontaux et verticaux, qui définissent son *4-voisinage*. Il s'agit des pixels de coordonnées $(x+1, y)$, $(x-1, y)$, $(x, y+1)$ et $(x, y-1)$. De façon formelle, deux pixels p_1 et p_2 sont 4-voisins, ou 4-adjacents, si $|x_{p_1} - x_{p_2}| + |y_{p_1} - y_{p_2}| = 1$. Quant aux quatre pixels voisins en diagonale de p, ils ont pour coordonnées $(x+1, y+1)$, $(x+1, y-1)$, $(x-1, y+1)$ et $(x-1, y-1)$. Ces quatre pixels, associés aux 4-voisins, forment les *8-voisins*. Ainsi, deux pixels p_1 et p_2 sont 8-voisins, ou 8-adjacents, si $max(|x_{p_1} - x_{p_2}|, |y_{p_1} - y_{p_2}|) = 1$.

On peut penser toute image numérique comme étant une fonction f ayant deux arguments discrets : la position spatiale des pixels, c'est-à-dire la largeur et la hauteur. Par convention, le pixel $(0, 0)$ se trouve en haut à gauche de l'image (la première coordonnée étant la largeur, la seconde la hauteur) ; pour une image ayant m pixels en largeur et n pixels en hauteur, les coordonnées des pixels varient donc de $(0, 0)$ à $(m-1, n-1)$. Cette fonction renvoie la couleur du pixel désigné :

$$f \ : \ \mathbb{N} \times \mathbb{N} \rightarrow \mathcal{C}$$

L'ensemble des couleurs possibles dépend du type d'images considéré (se référer à la figure 1.iii pour une illustration) :

et accessible à l'URL suivante : http://www.flickr.com/photos/arnolouise/3046105408/. Elle sera reprise tout au long de ce chapitre.

- $\mathcal{C} = \{noir, blanc\}$ pour les images binaires (en général, $noir = 0$ et $blanc = 1$) ;
- $\mathcal{C} = \mathbb{R}$ pour les images en niveaux de gris. Dans ce cas, seule l'intensité lumineuse est codée, en général sur 1 octet, avec 256 valeurs possibles (0 pour le noir, 255 pour le blanc) ;
- $\mathcal{C} = \mathbb{R}^3$ pour les images couleur (en général). La valeur de la couleur résulte alors d'une combinaison linéaire des valeurs de trois composantes.

FIG. 1.iii – Trois modes de représentation d'une même image : binaire (à gauche), niveaux de gris (au centre), couleurs (à droite, image originelle).

Résumons simplement quelques modes de codage informatique des couleurs. Le plus fréquent est le mode RVB (Rouge Vert Bleu, ou *RGB – Red Green Blue* – en anglais). Il se base sur la synthèse additive des couleurs (de façon identique à la lumière), et admet les trois couleurs primaires éponymes. Chacune a une valeur comprise entre 0 et 255, leur mélange permettant d'obtenir les couleurs intermédiaires (par exemple, [255,0,0] dénote le rouge, [255,255,0] le jaune, et [60,60,60] un gris foncé). Le mode de représentation des couleurs utilisé préférentiellement pour l'impression est CMJN (Cyan Magenta Jaune Noir, ou *CMYK – Cyan Magenta Yellow Key* – en anglais). Il se base sur la synthèse soustractive des couleurs (de façon identique à la peinture). Citons enfin le mode TSL (Teinte Saturation Luminance, ou *HSL – Hue Saturation Lightness* – en anglais), qui semble plus proche de la façon dont l'esprit humain appréhende les couleurs. Il repose sur un cercle des couleurs, dont l'angle donne la teinte, le taux de pureté étant quant à lui reflété par la saturation, et la luminosité par la luminance. Notons qu'il existe des formules permettant de passer d'un mode de représentation couleur à un autre.

La qualité visuelle d'une image numérique résulte en grande partie du nombre de pixels qui la composent, ce que l'on nomme couramment définition de l'image. Une notion connexe est celle de résolution. Il s'agit du nombre de pixels par unité de longueur. Plus celui-ci est élevé et plus la quantité d'information, et donc le degré de détail, sont importants. Ainsi, plus la résolution est élevée (le découpage est fin), et mieux la scène représentée l'est fidèlement (on dit couramment que l'image a une meilleur qualité). Notons qu'il est possible, et usité, de modéliser une même image à plusieurs résolutions sous la forme d'une pyramide [Burt et Adelson, 1983] (voir figure 1.iv). Cette pyramide peut ensuite être utilisée en traitement d'images pour travailler à différents niveaux de détail.

Nous pouvons rapprocher ces considérations sur la résolution du concept des images vignettes. Ces imagettes, de petite définition, qui sont très largement utilisées sur In-

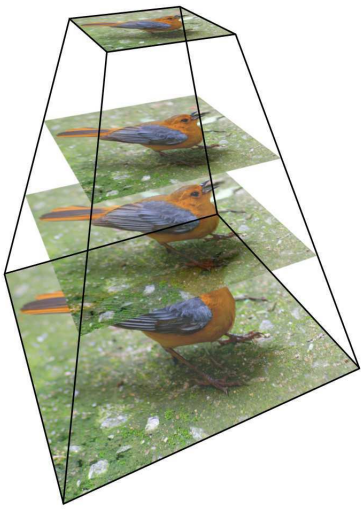


FIG. 1.iv – Une représentation multi-résolution d'une image : la pyramide.

ternet, résultent souvent du redimensionnement d'images originelles plus grandes. Ce redimensionnement engendre une perte d'information. Considérons pour illustrer nos propos la figure 1.v. L'image de gauche a vu sa définition réduite de moitié à résolution constante, comme on peut le voir sur l'image du centre. En l'affichant alors à sa dimension d'origine, on obtient l'image de droite : on constate clairement qu'une perte d'information a eu lieu. L'image vignette est donc un condensé d'information, selon un certain algorithme de redimensionnement, de l'image d'origine. Notons qu'il existe évidemment des méthodes d'interpolation pour tenter de permettre le retour à la dimension de l'image d'origine avec une perte minimisée.

FIG. 1.v – Création d'une image vignette et perte d'information engendrée.

On associe à une image numérique une taille, en octets, qui est fortement dépendante de la façon dont les informations sont représentées en mémoire. Il existe ainsi de nombreux formats, couleur ou non, propriétaires ou non, et avec ou sans compression (par exemple, JPEG, GIF, PNG, PPM...).

Lors de l'acquisition d'une image numérique, ou lors de traitements qui lui sont appliqués, de nombreux paramètres entrent en jeu. Il peut s'agir de conditions d'éclairage ou de luminosité, de changement d'échelle, de rotation, de l'existence de bruit, de l'occultation de certaines zones ou de recadrage, de modification de contraste... (voir la figure 1.vi pour une illustration). Ces éléments sont largement à prendre en compte pour les problématiques de reconnaissance de formes : l'enjeu est de parvenir à reconnaître un objet, en passant outre ces obstacles.

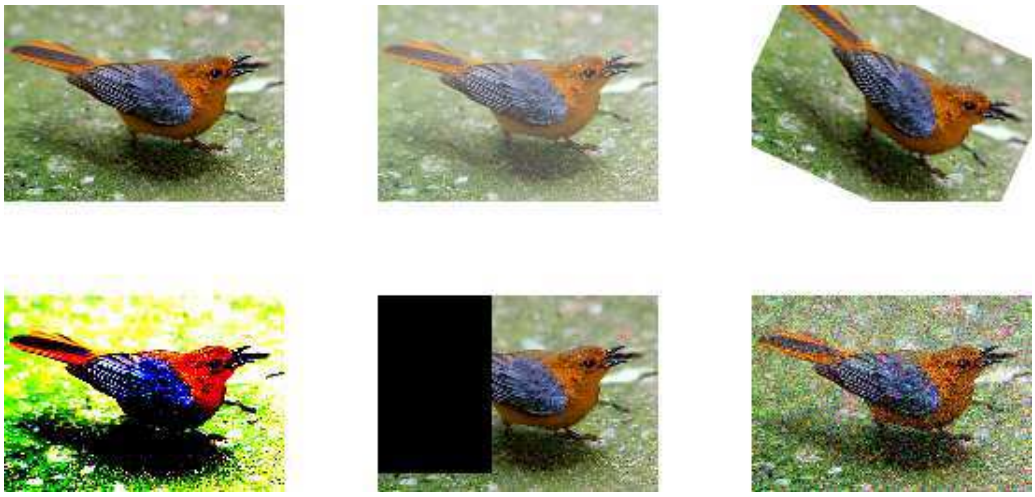


FIG. 1.vi – Différents traitements apportés à une image numérique. De haut en bas et de gauche à droite : image originelle, changement de luminosité, rotation et recadrage, modification de contraste, occultation, ajout de bruit.

Le contraste

La dernière notion que nous détaillerons est celle de contraste. Il s'agit d'une mesure de perception de la différence de luminosité entre deux zones de couleur. Plus sa valeur est élevée et plus grande est la différence. Nous avons déjà fait remarquer qu'une image couleur pouvait être codée selon trois composantes RVB (toute image couleur, binaire ou à niveaux de gris peut-être ramenée sur ce modèle – pour les deux derniers cas, chacune de ses composantes aura alors la même valeur). En réalité, il est également possible de représenter une image par trois autres valeurs (on peut passer à ce mode de représentation en appliquant de simples formules de conversion) :

- la luminance Y ;
- et deux valeurs de chrominance : U (différence de bleu) et V (différence de rouge).

Considérons une image ou une partie d'une image : il s'agit donc d'une matrice. En notant Y la luminance associée à un pixel, on peut calculer le contraste C associé à la matrice de nombreuses façons, dont les suivantes [Michelson, 1927, Kukkonen et al., 1993].

Contraste dit "simple" C

$$C = \frac{Y_{max}}{Y_{min}}$$

Contraste de Weber C_W

$$C_W = \frac{Y_{max} - Y_{min}}{Y_{min}}$$

Cette mesure est conseillée lorsque l'arrière plan est uniforme et de grande taille, avec un petit objet net au-dessus.

Contraste de Michelson C_M

$$C_M = \frac{Y_{max} - Y_{min}}{Y_{max} + Y_{min}}$$

Cette mesure est conseillée si l'arrière plan et le premier plan ne sont pas bien délimités, et se partagent équitablement l'image.

Contraste RMS C_{RMS} Soit n le nombre total de pixels constituant la matrice. Dans la suite, Y_p et $\bar{Y}$ correspondent respectivement à la luminance normalisée du pixel p de la matrice et à la moyenne des luminances normalisées des pixels de la matrice. Le contraste RMS est l'écart type des luminances des pixels :

$$C_{RMS} = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(Y_i - \bar{Y})^2}$$

Ces différentes définitions abordées, nous pouvons nous intéresser dans la partie suivante au traitement d'images numériques.

1.1.2 Deux opérations usuelles de traitement d'images

L'étude des images numériques et de leurs transformations, ou encore l'extraction d'information, sont souvent désignées par le terme traitement d'images. La frontière n'est pas toujours claire entre cette discipline et d'autres comme l'analyse d'images ou la vision par ordinateur. Il est admit que l'on peut distinguer trois types de traitement d'images [Gonzalez et Woods, 2006] :

- les traitements bas niveau : les entrées et les sorties sont toutes deux des images (c'était par exemple le cas des modifications illustrées sur la figure 1.vi) ;
- les traitements mi-niveau : les entrées sont des images, et les sorties des attributs extraits de ces images (c'est le cas, comme nous allons le voir, de la segmentation d'images et de l'extraction de caractéristiques) ;
- les traitements haut niveau : il s'agit de donner du sens à l'information dénotée par l'image, pour des applications telles que la reconnaissance de formes.

Nous nous intéresserons dans cette partie à deux opérations usuelles de traitement d'images de mi-niveau : la segmentation et l'extraction de caractéristiques. Ces deux opérations s'insèreront, par la suite, dans la problématique de la représentation d'images sous forme structurée.

Segmentation d'image

Un processus de segmentation vise à définir un ensemble de régions disjointes pour une image donnée, régions qui sont constituées de pixels voisins et homogènes selon un critère prédéfini. Soit R l'ensemble des pixels d'une image. Une segmentation partitionne R en n régions $R_1, R_2, \ldots, R_n$ telles que [Horowitz et Pavlidis, 1976, Gonzalez et Woods, 2006] :

- $\bigcup_{i=1}^{n} R_i = R$ (tout pixel appartient à une région) ;
- $R_i \cap R_j = \emptyset \; \forall i, j \in \{1, 2, \ldots n\} | i \neq j$ (les régions sont disjointes) ;
- R_i est un ensemble connexe $\forall i = 1, 2, \ldots, n$ (les pixels d'une région sont 4-adjacents ou 8-adjacents, selon des conditions prédéfinies) ;
- $\mathcal{Q}(R_i) = VRAI \; \forall i = 1, 2, \ldots, n$;
- $\mathcal{Q}(R_i \cup R_j) = FAUX$ pour tout couple de régions adjacentes R_i et R_j.

Pour les deux derniers points, $\mathcal{Q}$ désigne un prédicat logique défini sur les pixels des régions. Ainsi, tous les pixels d'une même région respectent au moins une propriété commune qui est le critère de la segmentation ; et les pixels de régions adjacentes diffèrent selon cette (ces) même(s) propriété(s).

Le challenge principal posé par la segmentation d'une image est celui du niveau de détail : idéalement, le processus de partitionnement doit s'arrêter lorsque les objets ou zones d'intérêt de l'image ont été détectés. Cette tâche est complexe, et de nombreuses approches existent [Bolon et al., 1995]. Parmi elles, l'approche frontière part du principe qu'au niveau des frontières des régions doivent se produire des changements d'intensité. On cherche alors à détecter ce qui peut s'assimiler à un contour. L'approche région tente elle un regroupement de pixels similaires. Il peut s'agir d'une modification d'une première partition (en divisant ou regroupant des régions), ou d'une croissance de régions en incorporant peu à peu de nouveaux pixels. Le résultat est évidemment largement dépendant de l'algorithme (et de ses éventuels paramètres) utilisé. De plus, comme on peut le voir sur la figure 1.vii, obtenue par un algorithme de *split and merge*[3], un même objet peut se retrouver segmenté différemment si ce qui l'entoure est modifié. Ici, le fond a changé, et les ronds orange, motifs de la tasse, ne sont pas toujours considérés comme des régions à part entière. Ce problème de sur et sous-segmentation est un des défis de cette opération de traitement d'images.

Extraction de caractéristiques

Une autre solution consiste à décrire l'image par un ensemble d'éléments caractéristiques de ses différentes parties. Ces éléments caractéristiques, qui sont des pixels ou des ensembles de pixels, sont bien souvent appelés points d'intérêt et ont été introduits pour la première fois par H. P. Moravec ([Moravec, 1977]). Dans l'idéal, ils se doivent d'être robustes à diverses modifications applicables à une image (comme déjà évoqué, il s'agit par exemple de changement de luminosité, de point de vue, de rotations...).

[3] Cette illustration a été obtenue en utilisant la version de démonstration de Descartes, un programme de segmentation interactive d'images disponible en ligne : http://www.greyc.ensicaen.fr/~luc/SEGMENTE/segmente.html

FIG. 1.vii – Une image de tasse avec deux fonds différents et les contours des régions correspondantes obtenues après segmentation.

De plus, ils doivent être suffisamment discriminants pour permettre l'identification des objets de l'image.

La première étape de cette approche est l'extraction des caractéristiques, et cela en utilisant un détecteur de points d'intérêt. Parmi les plus célèbres, on peut citer : Harris ([Harris et Stephens, 1988]) et Harris Affine ([Mikolajczyk et Schmid, 2004]), Susan ([Smith et Brady, 1995]), ou, plus récemment et de nos jours largement utilisé, SIFT ([Lowe, 1999, Lowe, 2004]). En règle générale, on estime qu'un bon détecteur doit être répétable et précis, détecter suffisamment de régions pour couvrir tous les objets de l'image, et fournir des régions riches en information. Plusieurs études ont été menées afin de comparer les détecteurs existants ([Schmid et al., 2000, Mikolajczyk et al., 2005]). La seconde étape est l'assignation d'information aux points détectés, par l'utilisation d'un descripteur. Celui-ci est également très important, et doit être répétable, compact et discriminant, et rapide à calculer. Il peut s'agir d'histogrammes (que nous détaillerons dans la partie 1.2.1), de moments, ou d'informations sur le voisinage des points (SIFT [Lowe, 2004], qui s'intéresse à l'orientation des gradients, est là aussi une référence).

Cependant, tout comme la segmentation, l'extraction de caractéristiques n'est pas exempte d'inconvénients. Intéressons-nous à la figure 1.viii : il s'agit d'une image de tasse, avec deux fonds différents[4]. Les flèches indiquent la position, l'échelle, et l'orientation des points d'intérêt. On remarque que le détecteur reste assez stable sur la tasse

[4]Cette illustration a été obtenue en utilisant la version de démonstration de SIFT, disponible en ligne : http://www.cs.ubc.ca/~lowe/keypoints/

elle-même, même si certains points ne sont pas détectés (certains détecteurs ont des résultats bien moins convaincants et ne détecteront que peu de points en commun). Par contre, il y a beaucoup plus de points détectés sur le fond herbe que sur le fond ciel. Il faut donc faire attention à la façon dont ces points seront utilisés par la suite, si l'on souhaite pouvoir reconnaître automatiquement qu'il s'agit de la même tasse, indépendamment du fond.

FIG. 1.viii – Utilisation du détecteur SIFT sur une même image de tasse avec deux fonds différents.

1.2 Premières représentations structurées d'images

Nous venons d'étudier des traitements bas niveau (rehaussement de contraste, changement de luminosité...) et mi-niveau (segmentation et extraction de caractéristiques) appliqués aux images numériques. Dans beaucoup d'applications, dont la reconnaissance de formes, la recherche ou la classification d'images, le but est de détecter automatiquement les objets représentés sur l'image, et d'y assigner éventuellement une signification. Il s'agit alors de traitement d'images haut niveau. Or, il est très difficile d'attribuer automatiquement un sens à une image si celle-ci est uniquement considérée par l'ordinateur comme étant une matrice de pixels. Il est donc nécessaire de passer à une autre représentation, que l'on appellera représentation structurée, puisqu'elle organise d'une façon ou d'une autre l'information contenue dans l'image. Le but est, dans l'idéal, double : parvenir à distinguer les différents objets de l'image et représenter les relations spatiales qui les lient. Pour cela, on n'a pas besoin de conserver toute l'information contenue dans

l'image d'origine, mais on doit préserver au mieux celle qui est importante.

Nous considérerons ici quatre formes de représentation, par ordre croissant de structuration : les histogrammes, les chaînes, les arbres et enfin les graphes. Cette partie n'est évidemment pas exhaustive, de nombreux autres modèles existent, comme l'approche par sacs de mots visuels qui a pour but de construire un vocabulaire sur les images ([Nowak et al., 2006]), qui peut par la suite être pondéré selon des critères de fréquence et de discrimination. On peut également citer les travaux portant sur les automates qui visent à définir des langages sur les images [Culik II et Kari, 1997, Kari, 2006] (ce qui est particulièrement bien adapté aux fractales) et peuvent aussi servir dans des buts de compression. Une autre approche est celle des squelettes [Blum, 1967], où les objets (préalablement détectés) d'une image sont représentés par un ensemble de points équidistants de sa frontière (il peut s'agir des centres des cercles de rayons maximaux inclus dans l'objet, mais plusieurs définitions mathématiques existent).

1.2.1 Représentation sous forme d'histogrammes

Les histogrammes sont parfois utilisés par les algorithmes de segmentation ou encore de compression d'images, mais peuvent également être considérés comme une représentation structurée – certes simple – à part entière [Swain et Ballard, 1991]. Un histogramme est une représentation alternative d'une image, de type statistique. C'est un diagramme en bâton représentant en ordonnées le nombre de pixels de l'image vérifiant la propriété de l'axe des abscisses. La figure 1.ix montre ainsi les trois histogrammes RVB d'une image.

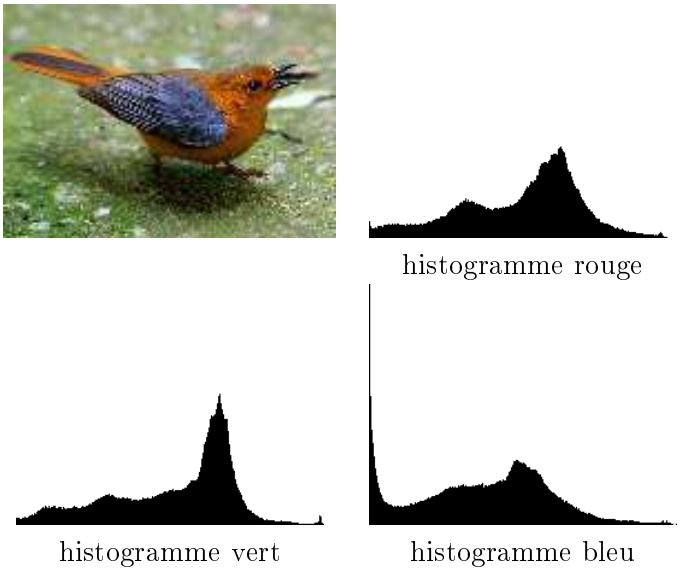


FIG. 1.ix – Trois histogrammes RVB d'une image. En abscisse : valeur de la composante (de 0 à 255), en ordonnées : nombre de pixels de cette valeur.

Il s'agit ici de reproduire la répartition des intensités des couleurs. Si l'on considère une image en niveaux de gris, si elle est sombre son histogramme sera concentré sur la gauche du graphique, inversement si elle est lumineuse il se trouvera à droite. Si cette image est fortement contrastée, l'histogramme sera bien réparti tout au long de l'axe des abscisses. On peut évidemment choisir d'autres informations que l'intensité des couleurs, pour obtenir des histogrammes plus riches et plus discriminants.

Une fois les histogrammes obtenus, encore faut-il pouvoir les comparer pour évaluer si les images dont ils sont les représentants sont semblables ou non. Plusieurs méthodes de calcul de distances entre histogrammes ont été développées dans ce but. Il peut s'agir de mesures d'intersection ([Swain et Ballard, 1990]), de distance euclidienne ou de distance quadratique [Smith et fu Chang, 1996]. Cependant, l'utilisation des histogrammes pour des tâches de reconnaissance de formes a des limites. Considérons la figure 1.x : elle représente deux images, qui ont exactement le même histogramme (la seconde image n'est en fait qu'une permutation des pixels de la première). Cet histogramme, qui représente les valeurs des pixels, n'est donc pas discriminant : une simple comparaison des distributions n'est pas adaptée à la perception visuelle. Pour pouvoir représenter la sémantique d'une image il faut donc utiliser au minimum un histogramme plus évolué. Une autre critique à l'encontre des histogrammes est la suivante : en tant que mesure globale, ils n'intègrent pas d'information spatiale sur la façon dont les objets de l'image sont agencés car ils ne permettent pas de les distinguer les uns des autres. Plusieurs autres approches ont depuis été développées, qui se basent sur des histogrammes locaux, ou sur des méthodes d'apprentissage [Chapelle et al., 1999].

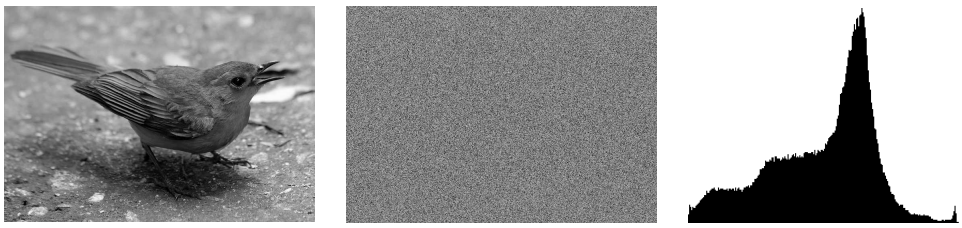


FIG. 1.x – Une limite des histogrammes : deux images de sémantique très différente ayant le même histogramme.

1.2.2 Représentation sous forme de chaînes

Une chaîne est une suite ordonnée d'éléments qui permet, de par ces données séquentielles, d'apporter des informations supplémentaires à celles fournies par exemple par les histogrammes. Étant donné une image, plusieurs méthodes ont été développées pour en extraire une représentation sous forme de chaîne. L'une d'entre elles, courante, se base sur l'utilisation des codes de Freeman [Freeman, 1961]. Il s'agit de parcourir le contour d'un objet de l'image, en codant sa forme spatiale. Ainsi, en partant du premier pixel de contour de l'objet rencontré en haut à gauche de l'image, et en visitant tour à tour ses voisins (ici on utilise couramment le 8-voisinage), on peut coder sous forme de

chaîne l'ensemble des directions prises. Ceci est illustré par la figure 1.xi qui représente le codage du chiffre 3. À chaque direction est associé un entier. La chaîne obtenue est la concaténation de ces entiers, dans l'ordre du contour. Ceci implique donc d'avoir auparavant détecté le contour des objets de l'image (ce qui a pu être mené à bien par une phase de segmentation).

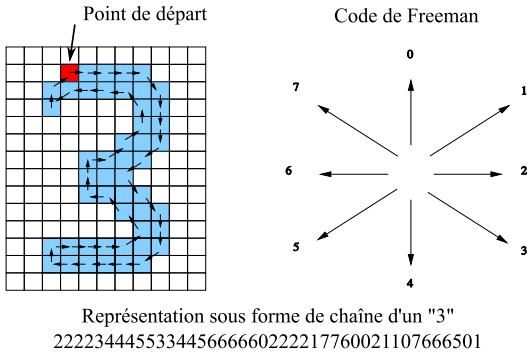


FIG. 1.xi – Représentation sous forme de chaîne de l'image d'un chiffre manuscrit, via le code de Freeman.

Une autre possibilité est d'extraire de l'image un ensemble de points d'intérêts, qui peuvent ensuite être ordonnés sous forme de chaîne selon un critère prédéfini. Dans [Ros et al., 2005, Solnon et Jolion, 2007], les auteurs proposent d'ordonner les points selon leur contraste, par ordre décroissant.

Une fois la représentation sous forme de chaîne obtenue, il s'agit d'être en possession de moyens pour les comparer. Outre la distance de Hamming, comptant le nombre de positions où deux chaînes diffèrent, la distance d'édition, introduite par Levenshtein [Levenshtein, 1966], définit des opérations sur les chaînes. On considère qu'il est possible de supprimer un élément de la chaîne, d'en insérer un, ou encore d'en substituer un par un autre. La suite d'opérations permettant de passer d'une chaîne à une autre est appelée script d'édition. Pour obtenir la distance d'édition entre deux chaînes, il faut trouver quel script d'édition est le moins coûteux. Pour cela, à chacune des opérations (insertion, suppression, substitution) est associé un coût, qui indique si la modification est bénigne ou non, et selon quel degré. La distance d'édition entre deux chaînes correspond alors au coût du script d'édition optimal. Cette approche a été utilisée dans [Bunke et Csirik, 1995, Solnon et Jolion, 2007], ces derniers ayant aussi étudié des distances entre histogrammes prenant en compte certaines données séquentielles.

Les chaînes trouvent aussi toute leur utilité lorsque l'on considère des classes d'images. En effet, d'autres mesures, telles que la chaîne médiane (généralisée ou non) ou la chaîne moyenne ont été définies [Bunke et al., 2002, Jiang et al., 2004]. Ces chaînes, qui minimisent la somme des distances aux chaînes de la classe, peuvent servir de représentantes de classes pour des travaux en classification supervisée. Pour savoir à quelle classe d'images appartient un nouvel exemple, il s'agit alors de le convertir en chaîne, puis de trouver de quelle représentante de classe il est le plus proche.

1.2.3 Représentation sous forme d'arbres

Un arbre est une représentation hiérarchique. Il s'agit d'une structure de données arborescente constituée d'une racine et de branches liant les nœuds entre eux, voie par laquelle la hiérarchie est dénotée. Si l'on choisit l'exemple d'un arbre généalogique, le départ de l'arbre, donc l'ancêtre dont on étudie la descendance, est appelé la racine ; ses descendants avec enfants sont les nœuds internes de l'arbre, et les descendants sans enfants sont appelés feuilles.

Les arbres peuvent être utilisés pour représenter les images de façon structurée. L'exemple le plus connu est celui des *quadtrees* [Finkel et Bentley, 1974, Samet, 1984]. Il s'agit d'arbres, basés sur un principe similaire à celui de *diviser pour régner*, construits de façon à ce que chaque nœud représente une partie de l'image initiale.

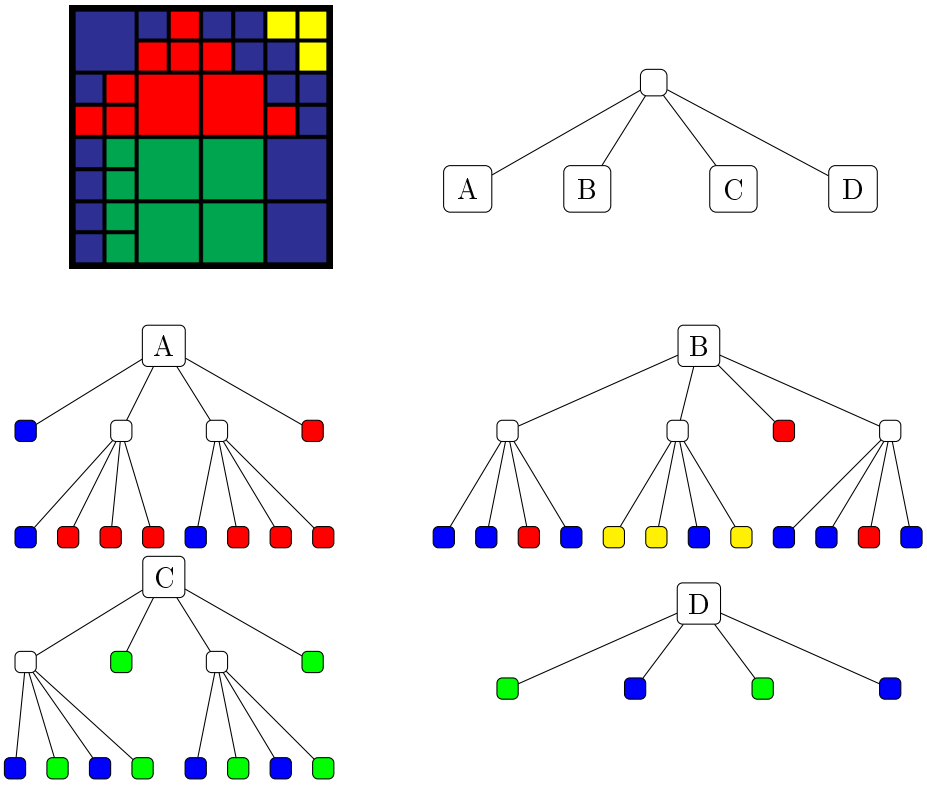


Fig. 1.xii – Une image simple et le quadtree correspondant.

Considérons la figure 1.xii, qui illustre le quadtree d'une image simple (l'arbre est ici scindé en quatre sous-arbres pour des besoins de visibilité). Il résulte de subdivisions successives de l'image : l'arbre débutant à la racine contient l'information de l'image complète. Chacun de ses quatre sous-arbres ordonnés représente un quart de cette image (A : quart en haut à gauche, B : quart en haut à droite, C : quart en bas à gauche, D :

quart en bas à droite). Tout nœud interne non homogène selon un critère prédéfini (ici, la couleur) est à son tour subdivisé en quatre fils. Pour finir, les feuilles de l'arbre sont étiquetées par leur valeur.

On se rend rapidement compte que la taille de l'arbre peut devenir très importante, et au pire des cas atteindre le nombre de pixels de l'image. Il existe pour pallier cela des méthodes permettant de le simplifier (on peut supprimer les fils d'un nœud s'ils sont homogènes à un certain seuil près, par exemple). Ceci conduit à des techniques de compression d'images si l'on reconstruit une image d'après le quadtree obtenu. De même, on peut également précéder la construction du quadtree par une étape de segmentation de l'image. Notons que d'autres partitionnements d'images sont possibles, ainsi les kd-trees [Bentley, 1975], les BSP trees (subdivision selon des axes arbitraires) [Qiu et Sudirman, 2001] et les arbres de partition [Salembier et Garrido, 1998] (représentation hiérarchique de régions de segmentation, permettant une vision à différentes échelles par fusion de nœuds).

Le code de Freeman, déjà évoqué pour la représentation d'images sous forme de chaînes, a également été adapté pour permettre une représentation sous forme d'arbres ([Bernard et al., 2008]). Comme l'illustre la figure 1.xiii, il s'agit en fait d'extraire la chaîne codant le contour de l'image. Celle-ci est, en parallèle, convertie en arbre : on crée une racine, d'étiquette -1, puis chaque nouveau code engendre un fils. Si ce code est répété plusieurs fois à la suite, des fils sont créés pour le nœud courant.

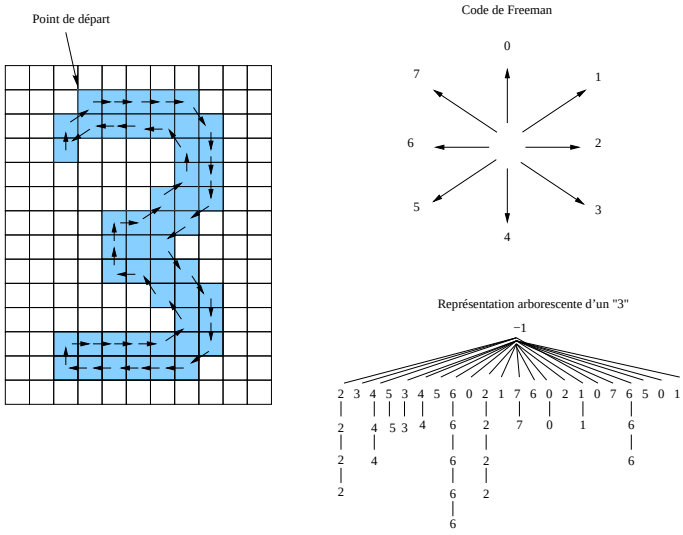


FIG. 1.xiii – Exemple d'une image d'un chiffre manuscrit représentée sous forme d'arbre, via l'utilisation du code de Freeman.

Étant donné des arbres représentant des images, il s'agit alors de pouvoir les comparer. Une possibilité est l'utilisation de la distance d'édition entre arbres. Déjà évoquée dans la partie précédente 1.2.2 sur les chaînes, la distance d'édition entre arbres admet principalement trois opérations [Bille, 2005] : la substitution du label d'un nœud par

un autre, la suppression d'un nœud interne (ses fils deviennent les fils de son père) et l'insertion d'un nœud. L'alignement d'arbres peut être considéré comme une variante de la distance d'édition [Jiang et al., 1994]. Il s'agit tout d'abord d'insérer des nœuds de labels vides jusqu'à ce que les deux arbres aient une structure identique, puis d'assigner un coût à cette suite d'opérations et aux substitutions de labels. On peut également citer les travaux sur l'inclusion d'arbres [Chen, 1998], un arbre étant inclus dans un autre si une suite de suppressions de nœuds permet de le retrouver.

1.3 Représentation structurée d'images sous forme de graphes

Des histogrammes aux arbres, en passant par les chaînes, nous avons gagné en structuration des données de l'image. Un modèle supplémentaire, plus complexe, permet d'accroître encore l'information codée : le graphe. Les graphes extraits d'images utilisés en reconnaissance de formes et vision par ordinateur sont dans la grande majorité des cas non orientés.

Définition 1.1 (Graphe non orienté)
Un graphe non orienté G est un couple (V, E) avec :
- *V un ensemble fini non vide d'éléments appelés sommets ou nœuds (*vertices *en anglais) de G ;*
- *$E \subseteq V \times V$ un ensemble fini d'arêtes (*edges *en anglais), qui sont des ensembles à deux sommets.*

Le graphe non orienté représenté sur la figure 1.xiv se définit en extension de la façon suivante :
- $V = \{A, B, C, D\}$;
- $E = \{\{A, B\}, \{B, C\}, \{C, C\}, \{C, D\}, \{B, D\}\}$.

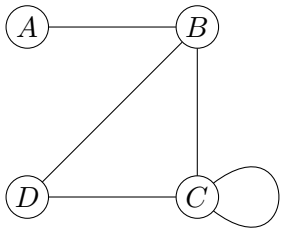


Fig. 1.xiv – Un graphe non orienté.

Nous verrons plus en détails le vocabulaire communément associé à la théorie des graphes dans la partie 3.1.1. Pour le moment, nous nous contenterons d'indiquer qu'une arête contenant deux fois le même sommet est appelée boucle (c'est le cas de l'arête $\{C, C\}$ sur la figure 1.xiv), et que la taille d'un graphe G, notée $|V|$, correspond à son

nombre de sommets. Par la suite, on désignera les graphes non orientés par le simple terme graphes.

Il est fréquent, et ce notamment lorsque les graphes représentent des images, de leur assigner des étiquettes, encore appelées labels (les nœuds du quadtree présenté dans la partie précédente étaient également étiquetés, par des informations de couleur). Appliquées aux sommets et/ou aux arêtes, ce sont des données qui permettent d'enrichir le graphe. Cela revient à ajouter à la définition générale d'un graphe deux fonction d'étiquetage :

Définition 1.2 (Graphe étiqueté)
Soient L_V et L_E deux ensembles d'étiquettes (respectivement pour les sommets et les arêtes de G). Un graphe non orienté étiqueté G est un quadruplet (V, E, α, β) avec :

- *V un ensemble fini non vide d'éléments appelés sommets ou nœuds (*vertices *en anglais) de G ;*
- *$E \subseteq V \times V$ un ensemble fini d'arêtes (*edges *en anglais), qui sont des ensembles à deux sommets ;*
- *$\alpha : V \to L_V$ la fonction d'étiquetage des sommets ;*
- *$\beta : V \times V \to L_E$ la fonction d'étiquetage des arêtes.*

1.3.1 Graphes d'images segmentées

Nous avons étudié la segmentation d'images dans la partie 1.1.2 et vu qu'elle permettait de subdiviser une image en régions. Il est possible, à partir de ce résultat, de construire des graphes de plusieurs façons (nous ne serons pas exhaustifs et nous contenterons d'introduire les principaux modèles).

Le premier modèle à avoir été défini est le RAG (*Region Adjacency Graph* en anglais) [Rosenfeld, 1974]. Il permet de décrire les relations d'adjacence entre les régions assez simplement. Illustrons nos propos par la figure 1.xv : il s'agit d'une image segmentée, et du RAG associé. Nous pouvons remarquer que chaque région est représentée par un sommet, et qu'il existe une arête entre deux sommets si les régions correspondantes sont adjacentes. Remarquons qu'un sommet supplémentaire a été ajouté : il correspond à une région infinie, qui représente l'extérieur de l'image, c'est-à-dire tout ce qui ne lui appartient pas. Sur l'illustration, nous avons choisi de labelliser les sommets par la couleur de la région qu'ils dénotent. En effet, pour cette catégorie de graphes dont les sommets représentent les régions, il est fréquent de les enrichir en indiquant des labels informatifs sur les régions : couleur, forme, surface, coordonnées du barycentre... pour s'adapter au mieux à l'image originelle. Il est également possible d'étiqueter les arêtes (par la longueur de la frontière commune entre les deux régions, par exemple).

On peut cependant faire plusieurs remarques négatives à l'encontre des RAG. La première est qu'ils ne permettent pas de différencier si des régions sont adjacentes ou incluses les unes dans les autres. De même, on ne peut pas savoir s'il existe une adjacence simple ou multiple entre elles. Enfin, étant donné que le modèle est simple, plusieurs images très différentes peuvent avoir le même RAG.

Une extension a depuis été proposée : il s'agit des graphes duaux [Kropatsch, 1994,

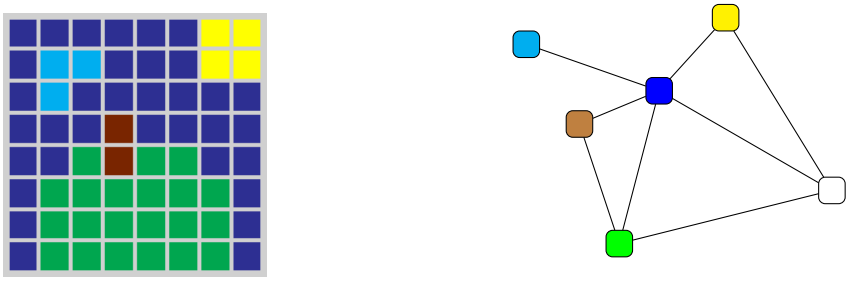


FIG. 1.xv – Une image simple et le RAG correspondant.

Kropatsch et Macho, 1995]. Il s'agit en fait de deux graphes, le premier étant une évolution du RAG, et le second son dual. Pour obtenir le dual G' d'un graphe G, il s'agit de choisir, pour chaque face de G (le dessin de G délimite l'espace en régions appelées faces, comme nous le verrons dans la partie 3.1.2), un point – points qui constitueront les sommets de G'. Pour chaque arête de G, on crée une arête dans G', qui relie les deux sommets associés aux deux faces séparées par l'arête de G. Le RAG devient également un multigraphe pour dénoter les adjacences multiples entre deux régions (c'est-à-dire que l'on peut créer plusieurs arêtes entre les deux mêmes sommets), et on introduit des boucles pour gérer les régions incluses (mais il faut alors être capable de détecter une boucle qui entoure géométriquement une autre région...). L'inconvénient des graphes duaux est qu'ils nécessitent de conserver deux graphes par image, mais aussi et surtout qu'il faut définir géométriquement les arêtes et les boucles.

Par la suite, d'autres évolutions ont vu le jour, et ont mené à l'introduction des cartes combinatoires, qui permettent de représenter toutes les relations d'adjacence et d'incidence entre les régions [Edmonds, 1960, Tutte, 1963, Cori, 1975]. Nous nous limitons dans cette thèse aux images en deux dimensions, mais notons que les cartes combinatoires peuvent être définies en dimension quelconque.

Une carte combinatoire peut être vue comme une structure de données représentant chacune des faces d'un graphe. Elle est constituée d'un ensemble de brins qui correspondent aux arêtes des faces (toute arête participant deux fois pour définir les faces d'un graphe, il existe deux fois plus de brins que d'arêtes). On définit, sur ces brins, une permutation (il s'agit d'une bijection de l'ensemble des brins dans lui-même). Celle-ci permet de parcourir chaque face et représente donc la relation d'adjacence entre les arêtes. Par la suite, les faces ainsi représentées sont mises en relation par une involution (qui est une bijection b de l'ensemble des brins dans lui-même, telle que $b = b^{-1}$) qui permet de représenter l'adjacence entre les faces. Plus formellement, on a la définition suivante [Lienhardt, 1991] :

Définition 1.3 (Carte combinatoire 2D)
Une carte combinatoire 2D $M = (D, \beta_1, \beta_2)$ est définie par :

	1	2	3	4	5	6	7	8
β_1	2	3	4	5	6	7	1	9
β_2	15	14	18	17	10	9	8	7

TAB. 1.1 – Données de β_1 et β_2 pour la carte combinatoire 2D de la figure 1.xvi.

- *un ensemble de brins D ;*
- *une permutation β_1 sur D, permettant de définir les faces ;*
- *une involution β_2 sur D, permettant de changer de face.*

Considérons ainsi la figure 1.xvi représentant une carte combinatoire en dimension 2. Les données associées à β_1 et β_2 sont explicitées sur le tableau 1.1.

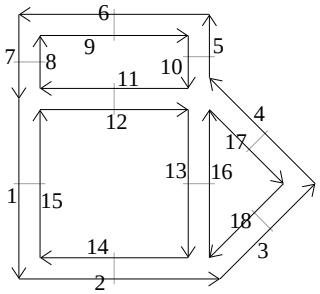


FIG. 1.xvi – Une carte combinatoire 2D.

Les cartes combinatoires offrent la possibilité de retrouver, par les fonctions β_i, les faces ou les sommets des graphes qu'elles représentent. De plus, elles peuvent être accompagnées de structures d'inclusion (rendant compte de l'inclusion de régions les unes dans les autres), dont il existe plusieurs types.

1.3.2 Graphes et points d'intérêt

D'autres méthodes de construction de graphes à partir d'images ne nécessitent pas de phase préalable de segmentation. À la place, elles font appel à un détecteur de points d'intérêt (voir la partie 1.1.2 sur l'extraction de caractéristiques), qui vont ensuite être reliés par des arêtes pour former un graphe. Une méthode communément admise pour cela est de créer des graphes de voisinage. Il peut s'agir :

- de l'arbre couvrant de longueur minimale $\mathcal{EMST}$ (*Euclidean Minimum Spanning Tree*), dont la somme des longueurs euclidiennes des arêtes est minimisée (on peut voir un arbre comme étant un graphe particulier) ;
- du graphe des voisins relatifs $\mathcal{RNG}$ (*Relative Neighbourhood Graph*) pour lequel il existe une arête entre deux sommets v_1 et v_2 si l'intersection des cercles de centre v_1 et v_2 et de rayon v_1v_2 ne contient pas d'autre sommet ([Toussaint, 1980]) ;

- le graphe de Gabriel $\mathcal{GG}$ (*Gabriel Graph*) pour lequel il existe une arête entre deux sommets v_1 et v_2 si le cercle de diamètre v_1v_2 ne contient pas d'autre sommet ([Gabriel et Sokal, 1969]) ;
- le graphe de Delaunay, ou encore la triangulation de Delaunay $\mathcal{T}$, dont les propriétés sont expliquées en détail dans la partie A. Il s'agit d'une subdivision planaire dont les faces sont des triangles, et telle que le cercle circonscrit à tout triangle ne contienne aucun autre sommet.

On peut remarquer que les arêtes de ces graphes de voisinage vérifient [Fortune, 1992] $\mathcal{EMST} \subseteq \mathcal{RNG} \subseteq \mathcal{GG} \subseteq \mathcal{T}$ (pour une illustration, voir la figure 1.xvii[5]).

Les triangulations sont fréquemment utilisées pour représenter des images en deux dimensions ([Kohout, 2007]) ou pour modéliser des objets en 3D ([Garcia et al., 1999]), en intégrant, si ce ne sont des points d'intérêt, au moins des pixels dits significatifs. On parle alors souvent de maillages. Ces modélisations sont aussi un moyen de compresser et approximer les images ([Taubin et Rossignac, 1998, Rossignac, 1999, Alliez et al., 2008]).

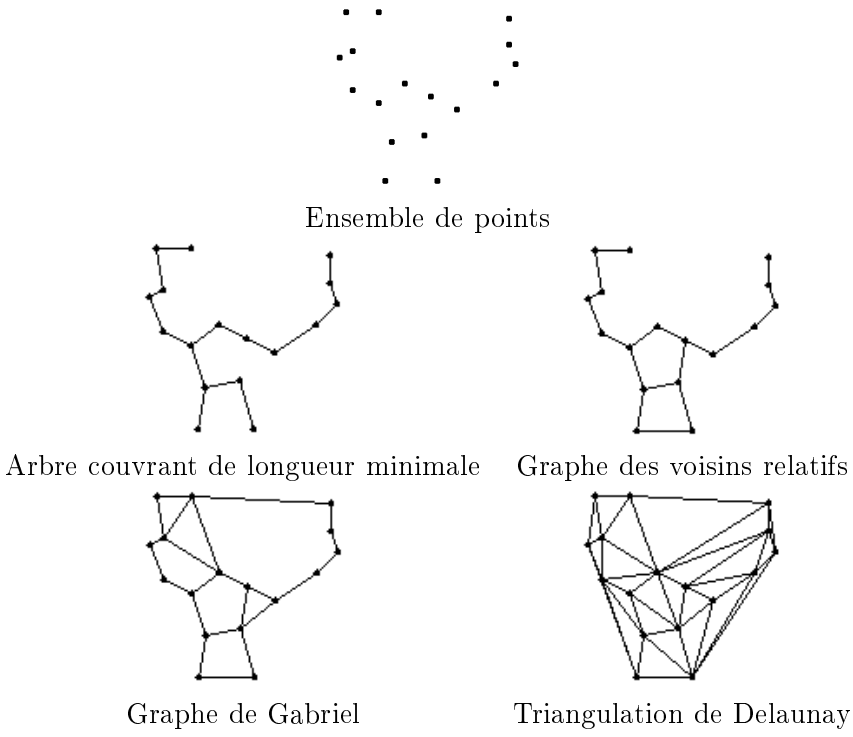


FIG. 1.xvii – Différents graphes de voisinage d'un même ensemble de points.

Comme pour les graphes issus de segmentation, l'étiquetage des sommets et arêtes est possible. Cependant, ceux-ci n'ont plus du tout la même signification. Un sommet est

[5]Cette illustration a été obtenue en utilisant l'applet suivante disponible en ligne : http://www.cim.mcgill.ca/~sveta/cg/applet1.html

une région d'un côté, et un ou plusieurs pixels significatifs de l'autre. Une arête dénote une adjacence de régions pour l'un, et une notion de plus proche voisin de l'autre. L'étiquetage des sommets est confié au descripteur de points d'intérêt, celui des arêtes est encore à étudier.

De façon analogue aux histogrammes, chaînes et arbres, une fois les graphes obtenus, il faut pouvoir les comparer. C'est là tout l'objet des problématiques d'isomorphisme de graphes, que nous étudierons en détail dans le chapitre 3.

2

Vers une fonction de coût d'appariements de graphes

Sommaire

Considérons que nous sommes en possession de graphes représentant des images (plusieurs approches pour en obtenir ont été évoquées dans le chapitre 1 et nous présenterons notre méthode d'extraction dans le chapitre 4). Pour répondre à des besoins en reconnaissance de formes ou en classification, nous avons besoin de comparer ces images. Cette tâche se ramène à l'étude des graphes les représentant et à la découverte de caractéristiques communes ou différentes. Il s'agit en fait d'assigner telle(s) partie(s) d'un graphe à telle(s) partie(s) d'un autre graphe. Une telle association, qui portera le nom d'appariement de graphes, devra respecter certains critères et commettre un minimum d'erreurs, tout en assurant un certain degré de liberté.

Ainsi, ce chapitre porte sur la mise en place d'une fonction de coût sur les appariements qui doit mesurer leur qualité et qui, une fois définie, devra permettre de les

optimiser. Nous commencerons par présenter les principes qui régissent les appariements de graphes, les concepts de qualité et coût associés, et nous montrerons qu'il s'agit d'un problème combinatoire. Puis, nous porterons notre attention sur les critères permettant de définir une fonction de coût d'appariements de graphes. Nous verrons que ces critères sont liés aux graphes, mais aussi intimement au domaine de l'image (descripteurs attachés aux sommets). Nous présenterons enfin un compte rendu des expérimentations que nous avons menées pour analyser notre fonction. En particulier, nous discuterons les idées et variantes que nous avons imaginées.

Malgré nos efforts, nous n'avons pas obtenu les résultats escomptés, ce qui nous a conduit à remettre en cause notre approche. En effet, nos résultats se sont avérés décevants, ce qui est en partie dû à notre inexpérience dans le domaine de l'image, mais également à la multiplicité des critères que nous avons cherché à optimiser. Aussi, dans les chapitres ultérieurs, nous avons réorienté nos travaux afin de nous concentrer sur un seul de ces critères : le respect de la structure des graphes.

2.1 Appariements de graphes

Commençons par étudier les différents types possibles d'appariements de graphes, et demandons-nous si, étant donné un de ces appariements, il existe un moyen de juger de sa pertinence.

2.1.1 Définitions

Un appariement de graphes (*graph matching* en anglais) est une relation liant deux graphes, consistant en la mise en correspondance des sommets du premier avec ceux du second.

Définition 2.1 (Appariement de graphes)
Soient $G_1 = (V_1, E_1)$ et $G_2 = (V_2, E_2)$ deux graphes. Un appariement de G_1 et G_2 est une relation $r \subseteq V_1 \times V_2$ telle que, si $(v_1, v_2) \in r$, alors v_1 est apparié à v_2.

Aucune contrainte *a priori* n'est fixée sur les graphes : ils peuvent donc avoir une taille différente. De même, rien n'oblige à ce que tous les sommets soient appariés. On distingue ainsi plusieurs types d'appariements, en fonction des contraintes imposées (voir également les illustrations de la figure 2.i) :

- appariement bijectif : tout sommet de G_1 (respectivement G_2) est apparié à exactement un sommet différent de G_2 (respectivement G_1). Les deux graphes ont donc la même taille, et la cardinalité de la relation est $(1, 1)$;
- appariement injectif de G_1 dans G_2 (cardinalité $(1, 0/1)$) : tout sommet de G_1 est apparié à exactement un sommet différent de G_2 (on a donc $|V_1| \leq |V_2|$) ;
- appariement univoque, de cardinalité $(0/1, 0/1)$: chaque sommet apparié de G_1 et G_2 l'est avec au plus un sommet ;
- appariement multivoque, de cardinalité $(0 \ldots |V_2|, 0 \ldots |V_1|)$: aucune contrainte particulière n'est imposée, chaque sommet peut être apparié à zéro, un ou plusieurs sommets.

Appariement bijectif

Appariement injectif

Appariement univoque

Appariement multivoque

FIG. 2.i – Types d'appariements (les couleurs dénotent les sommets appariés. Les sommets blancs sont non appariés).

La notion d'appariement de graphes est intimement liée à celle d'isomorphisme de graphes, et nous étudierons différentes méthodes pour les obtenir dans le chapitre 3.

2.1.2 Concepts de qualité et coût

En présence d'un appariement potentiel entre deux graphes, il faut pouvoir juger de sa pertinence, dans le but de trouver le meilleur possible, de rejeter un mauvais appariement, ou de l'améliorer.

Pour mettre en place un moyen de juger de sa qualité, il est nécessaire de définir ce que nous appelons le coût d'un appariement, critère qui doit permettre de discriminer un bon appariement d'un moins bon : son coût sera plus faible. Sur quoi ce coût peut-il se baser ? Sur des contraintes que l'appariement doit respecter. Ces contraintes peuvent être dures, c'est-à-dire qu'il doit être obligatoire de les respecter, ou souples, c'est-à-dire que l'appariement doit les satisfaire au mieux. Le rôle de la fonction de coût est d'assigner des pénalités aux contraintes dures non respectées, mais aussi aux contraintes souples, en fonction de leur degré de satisfaction. Dans le cadre des graphes, intuitivement, ces contraintes se portent sur les sommets (Est-il juste d'apparier tel et tel sommets ? Cela importe-t-il si je n'apparie pas tel autre sommet ?) et sur les arêtes (Si j'apparie ces deux sommets à ces deux autres, l'arête qui existe entre eux dans le premier graphe existe-t-elle aussi dans le second ?). Interviennent alors les étiquettes assignées aux sommets et/ou aux arêtes.

2.1.3 Un problème combinatoire

Comme nous venons de l'indiquer, la fonction de coût doit être discriminante : un bon appariement aura un coût plus faible qu'un appariement de moins bonne qualité. Ainsi, plus la qualité de l'appariement est élevée et plus son coût diminue. L'appariement optimal aura donc un coût minimal. Un appariement parfait (c'est-à-dire que toutes les contraintes dures et souples auront été parfaitement respectées) aura quant à lui un coût de 0. Ces considérations portent à analyser les graphes en tant qu'objets combinatoires : parmi tous les appariements possibles, quel est celui qui minimise le coût ? Comment l'obtenir ? On se ramène alors à un problème d'optimisation : trouver $r \subseteq V_1 \times V_2$ qui minimise le coût.

S'il n'est pas raisonnablement possible d'obtenir un très bon appariement de façon directe, il peut être néanmoins envisageable, à partir d'un appariement de base (obtenu par des méthodes ayant pour but de fournir un appariement vraisemblable à l'aide d'heuristiques) et du calcul de son coût, d'effectuer des modifications sur l'appariement (désappariements de sommets, échanges d'appariements, création de nouveaux appariements...) afin de diminuer son coût et d'obtenir, à terme, un appariement, s'il n'est optimal, tout du moins suboptimal. On peut alors faire appel à des techniques d'optimisation, les approches par voisinage ([Kwang Y. Lee, 2008]). Citons parmi elles les algorithmes génétiques et la recherche locale ([Emile Aarts, 2003]). Il s'agit d'explorer par voisinage l'espace des combinaisons d'un problème d'optimisation combinatoire, dans le but de trouver des solutions parmi les meilleures. L'idée est la suivante : à partir d'une,

ou de plusieurs, combinaisons initiales, on se propose de générer à chaque étape une ou plusieurs combinaisons voisines des combinaisons courantes. Les algorithmes génétiques s'inspirent de la théorie de l'évolution en biologie, qui dénote les transformations subies par les espèces vivantes au fil des générations, et qui explique la diversification des formes de vie. Les algorithmes génétiques prennent en considération trois mécanismes évolutifs : la sélection naturelle (les organismes les mieux adaptés à leur environnement ont la plus grande probabilité de survie), les réarrangements chromosomiques liés à la reproduction par croisements et les phénomènes de mutation. L'idée est de faire évoluer un ensemble de combinaisons selon ces trois mécanismes, en cherchant à optimiser leur capacité d'adaptation à leur environnement, ce qui correspond à la fonction que l'on cherche à optimiser. Pour la recherche locale, il s'agit de partir d'une combinaison initiale et d'explorer son voisinage de proche en proche, en sélectionnant à chaque étape une nouvelle combinaison. On considère donc qu'une combinaison donnée possède un ensemble de voisins obtenables par des modifications élémentaires (changement de valeur, échange de valeurs...). L'opérateur de voisinage est déterminant pour les résultats de l'algorithme, et devrait permettre, quelque soit le point de départ, d'obtenir la meilleure solution. Ainsi, les travaux présentés dans [Petrovic et al., 2002, Sorlin et Solnon, 2005] utilisent la recherche tabou pour améliorer des appariements fournis par une méthode gloutonne : il s'agit d'explorer les solutions du voisinage, tout en gardant en mémoire les dernières modifications effectuées, afin de ne pas boucler sur des optima locaux.

Enfin, citons également les approches constructives qui ont pour but de construire incrémentalement des solutions en utilisant en général un modèle stochastique pour choisir à chaque étape le prochain élément. Parmi elles, l'optimisation par colonies de fourmis ([Solnon, 2008]) s'inspire du comportement collectif de ces insectes : la probabilité de choix d'un élément est proportionnelle à la quantité de phéromone déposée précédemment. Celle-ci évolue à la fois par un mécanisme de renforcement pour les meilleures combinaisons, et d'évaporation pour privilégier la nouveauté.

2.2 Critères de qualité

Nous nous proposons dans cette partie de détailler l'élaboration d'une fonction de coût d'appariements de graphes. À cet effet, nous avons considéré les trois niveaux de coût suivants :

1. **le niveau du point**. On cherche ici à répondre aux questions suivantes : que coûte le fait d'apparier tel sommet avec tel autre ? Que coûte le fait de ne pas apparier tel sommet ?
2. **la structure du graphe**. Ce niveau de coût s'intéresse à la conservation des arêtes des graphes. Que coûte le fait d'apparier deux sommets d'un graphe à deux d'un autre graphe, alors qu'il existe une arête entre eux dans le premier graphe, mais pas entre leurs images dans le second ?
3. **le maintien de l'ordre**. Il est question de l'ordre de contraste des sommets, et de la préservation de cet ordre par l'appariement. Que coûte le fait d'apparier un sommet très contrasté à un moins contrasté ?

Ces critères partent de plusieurs constats. Premièrement, il semble évident que les descripteurs associés aux points d'intérêt entrent en jeu lorsque l'on apparie deux sommets. Pour autant, le but n'est pas de mettre au point un descripteur le plus performant possible : ce n'est pas notre domaine, et nous souhaitons nous focaliser sur le graphe, qui risquerait d'être masqué si le descripteur seul suffisait à apparier les points entre eux. Nous verrons donc que nous choisirons un descripteur simple. Deuxièmement, vérifier le respect de la structure du graphe est tout aussi logique : tout l'apport des graphes réside dans l'existence d'arêtes entre les sommets, qui sont une source d'information. D'autres travaux ont ainsi intégré le respect de la structure, comme dans [Ta, 2010] où les auteurs vérifient la cohérence des angles de rotation avec les sommets adjacents lors de l'appariement. Enfin, le troisième niveau de coût se base sur le contraste, car c'est une donnée qui nous a semblé entrer souvent en jeu lors de l'extraction de caractéristiques. Ce critère est lié au domaine de l'image, mais des travaux antérieurs l'ont utilisé dans le cadre de calculs de distances entre représentations structurées d'images, et semblent en avoir amélioré les résultats [Solnon et Jolion, 2007].

Intéressons-nous aux graphes que nous allons utiliser. Dans le cadre de notre travail, aucune contrainte n'est fixée *a priori* sur ce que représentent les images, ni sur le type de fichier considéré (qu'il soit en noir et blanc, en niveaux de gris, en couleur, et quelque soit son extension). Cependant, nous ne considérerons que le cas des appariements univoques. De plus, nous nous plaçons dans le cadre d'images dites vignettes (introduites dans la partie 1.1.1), images que l'on trouve fréquemment sur Internet, par exemple lors d'une recherche d'images sur un moteur de recherche. Leur taille en pixels et leur résolution sont donc faibles. Nous partons du postulat que les techniques de segmentation ne sont alors pas totalement adaptées (les régions risquant d'être mal définies de par la faible définition). Nous avons donc choisi de baser la création des graphes sur l'extraction de points d'intérêt, et de les relier par la triangulation de Delaunay, qui fournit un graphe de voisinage riche, tout en étant planaire (voir l'annexe A pour des informations sur la triangulation de Delaunay, et la partie 3.1.2 pour des détails sur les propriétés des graphes planaires). Ces arêtes ne seront pas étiquetées, afin de traiter des graphes *épurés* dans un premier temps.

Le détecteur de points d'intérêt utilisé [Bres et Jolion, 1999] se base sur la notion de contraste. Cette méthode s'appuie sur une représentation multirésolution de l'image considérée sous la forme d'une pyramide (voir la partie 1.1.1), avec rehaussement itératif du contraste à tous les niveaux de résolution. Il s'en suit une reconstitution de l'image rehaussée sous la forme d'une image binaire, qui conserve au mieux l'information de contraste tout en réduisant l'entropie de l'image de départ. Les points d'intérêt sont alors les maxima locaux de la mesure de contraste cumulée dans la base de la pyramide. Sont retenus les pixels en question, et leur voisinage immédiat V_8 en binaire, que l'on appellera masque (les valeurs possibles des pixels d'un masque sont donc 0 : pixel noir ou 1 : pixel blanc). Il existe ainsi, selon la taille de ce voisinage, $2^9 = 512$ masques possibles. Des exemples de masques sont présentés sur la figure 2.ii. On conserve également pour chaque point sa valeur de contraste, réel compris entre 0 et 1, qui exprime, si ce n'est une forte information séquentielle, au moins un degré d'importance.

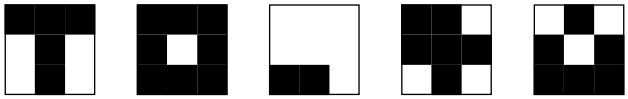


FIG. 2.ii – Exemples de masques 3 × 3 de points d'intérêt

2.2.1 Apparier deux sommets : le niveau du point

Nous pouvons ici distinguer deux cas : deux points ont été appariés, ou un point n'a pas été apparié.

Lorsque deux points sont appariés

Nous avons mentionné que tout point d'intérêt était étiqueté par son masque. Lorsque deux sommets sont appariés, il est donc nécessaire de pouvoir calculer si les deux masques sont proches l'un de l'autre ou non. Il s'agit ainsi de définir une mesure de distance entre masques (il est clair que la taille du voisinage des masques peut être modifiée, pour prendre en compte un voisinage plus étendu si nécessaire).

Distance de Hamming Étant donné deux suites de symboles de même longueur, la distance de Hamming d_H [Hamming, 1950] se définit comme le nombre de positions pour lesquelles les symboles ne sont pas les mêmes. Ainsi, si l'on considère des suites binaires, on a le résultat suivant : $d_H(01101, 00111) = 2$. De même, pour des chaînes de caractères : $d_H(\text{"}tasse\text{"}, \text{"}pacte\text{"}) = 3$. Cette distance peut s'appliquer aux masques binaires aisément, en comptant le nombre de pixels qui diffèrent d'un masque à l'autre. Alors :

$$d_H(\ \blacksquare\blacksquare\blacksquare\blacksquare\ ,\ \blacksquare\blacksquare\blacksquare\blacksquare\ ) = 2$$

Pour les masques, d_H est un entier compris entre 0 (tous les pixels sont identiques) et 9 (tous les pixels sont différents). La normalisation est aisée pour obtenir un réel compris entre 0 et 1. Notons que la distance de Hamming est une métrique. En effet, quelque soit la nature de l'ensemble X, $d_H : X \times X \to \mathbb{R}$ vérifie :

- $d_H(x, y) \geq 0 \forall x, y \in X$: non-négativité ;
- $d_H(x, y) = 0 \forall x, y \in X$ si et seulement si $x = y$: unicité ;
- $d_H(x, y) = d_H(y, x) \forall x, y \in X$: symétrie ;
- $d_H(x, y) \leq d_H(x, z) + d_H(z, y) \forall x, y, z \in X$: inégalité triangulaire.

La distance de Hamming sur les masques peut être adaptée, si l'on considère que le fait d'appliquer certaines transformations aux masques (par exemple, rotation ou symétrie) ne doive pas être pris en compte par la mesure de distance. Lors du calcul de d_H, on peut donc choisir d'être aux rotations près ou aux symétries près.

Classes de masques Outre la distance de Hamming, il est possible de définir des classes de masques, qui seraient déterminées par rapport à la structure formée par leurs

pixels. Une classification en 100 classes a ainsi été proposée [Jolion et Simand, 2004], caractérisant des coins, des zones uniformes, des contours à 45 degrés... Les distances entre masques peuvent alors être apprises sur un jeu d'exemples. Le principe est le suivant : ordonnons les masques de deux images sous forme de chaîne, par ordre de contraste décroissant. Étant donné deux chaînes dont on extrait un couple de masques à une position donnée, si les deux images appartiennent à la même classe alors la distance entre les deux masques est diminuée d'un certain alpha, sinon elle est augmentée. Le processus est répété jusqu'à convergence des distances ou jusqu'à un nombre fixe d'itérations. Plus simplement, on peut aussi considérer que si deux masques font partie de la même classe alors ils ont une distance de 0, et une distance de 1 sinon.

Lorsqu'un point n'est pas apparié

Le fait de pénaliser les sommets que l'on apparie si leurs masques diffèrent ne doit pas induire en erreur la fonction de coût. En effet, il suffirait alors de n'apparier aucun sommet pour obtenir un coût minimal au niveau du point. Pour éviter cet effet indésirable, il nous faut également sanctionner les points non appariés.

Il s'agit de s'interroger sur le coût des points non appariés : de nouveau, nous allons nous intéresser à leurs masques, et plus précisément à l'existence de masques qui, de par la disposition de leurs pixels, seraient plus importants que d'autres, et pour lesquels le coût de non appariement doit être plus élevé. On se demande alors s'il existe un ou plusieurs masques dits "moyens", aux rotations près, à partir desquels on pourrait calculer le coût des masques non appariés.

En pratique, en utilisant comme distance entre masques le minimum de la distance de Hamming aux rotations près, on obtient les masques moyens (minimisant la somme des distances aux autres masques) de la figure 2.iii. Celle-ci contient 32 masques, nombre réduit à 8 aux rotations près. Ces masques moyens sont à une distance minimale moyenne de Hamming aux rotations près de 2.9 de tous les autres masques.

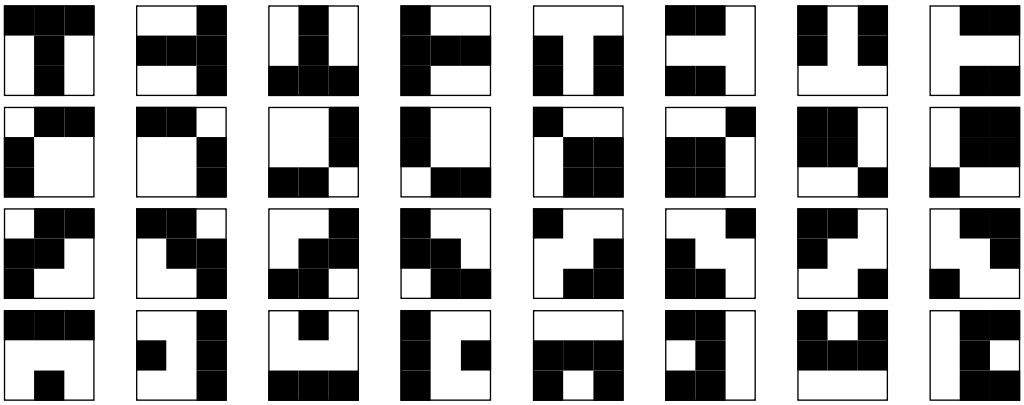


FIG. 2.iii – Masques moyens selon la distance de Hamming aux rotations près

Il est désormais possible de calculer le coût de non appariement d'un masque en lui

assignant la plus petite distance de Hamming (à certaines transformations près si nécessaire), entre lui-même et l'ensemble des masques moyens. Cette distance peut ensuite être normalisée, afin d'obtenir une valeur comprise entre 0 et 1 (la borne supérieure n'étant jamais atteinte).

2.2.2 Apparier deux graphes : le respect de la structure

Le second point que nous souhaitons aborder est celui de la conservation des arêtes lors de l'appariement. En effet, il ne s'agit pas seulement de faire coïncider les masques entre eux. Il faut également que la configuration spatiale des points d'intérêt soit prise en compte. Celle-ci est directement reflétée par la triangulation de Delaunay utilisée pour relier les points entre eux. Deux graphes de la même classe doivent donc avoir une triangulation assez proche. Lors de l'appariement, on va alors vérifier si deux sommets adjacents dans le premier graphe sont appariés à deux sommets adjacents dans le second.

Une première possibilité est donc de pénaliser toute arête non directement conservée par l'appariement par un coût de 1. Néanmoins, même si une arête peut ne pas être directement préservée, elle peut tout de même l'être par une chaîne (c'est-à-dire une suite d'arêtes consécutives, dont le nombre définit la longueur – cette notion sera définie dans la partie 3.1.1). Le coût de la structure du graphe peut donc être amélioré, en s'intéressant à la façon dont la connexité est préservée. La non préservation d'une arête aura alors un coût pondéré par la longueur de la chaîne qui la remplace (la triangulation de Delaunay étant connexe, il existe une chaîne entre toute paire de sommets).

Ce niveau de coût implique donc le calcul préalable de la fermeture transitive des deux graphes étudiés (cela revient à calculer la longueur de la chaîne entre toute paire de sommets de chacun des graphes). Celui-ci peut se réaliser en $\mathcal{O}(n^3)$ avec l'algorithme de Roy-Warshall [Cormen et al., 1990]. Pour chaque arête non directement conservée par l'appariement mais remplacée par une chaîne de longueur l, le coût de la structure du graphe c_{sg} est augmenté de :

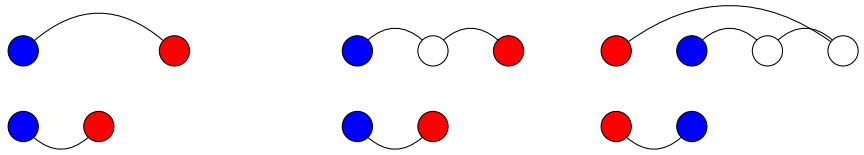
$$c_{sg} += ln(|1 - l|)$$

Ainsi, comme on peut le voir sur la figure 2.iv où les arêtes existantes dans chacun des deux graphes sont représentées par des arcs de cercle et où les appariement sont dénotés par des couleurs :

- associer une arête à une chaîne de longueur 2 coûte $ln(|1-2|) = ln(1) = 0$: nous n'introduisons aucune pénalisation, pour parer par exemple à du bruit ou de l'instabilité dans la détection des points d'intérêt ou sur l'image, ou pour compenser le fait que nous ne considérons que des appariements univoques ;
- associer une arête à une chaîne de longueur 3 coûte $ln(|1-3|) = ln(2) = 0.69$;
- associer une arête à une chaîne de longueur 4 coûte $ln(|1-4|) = ln(3) = 1.1$;
- ...

2.2.3 Intégrer la notion de contraste

Dans l'introduction de cette partie, nous avons évoqué le fait qu'à tout masque était associée sa valeur de contraste. Cette donnée est la base d'un postulat que nous



$c_{sg} = 0$
l'arête est directement préservée

$c_{sg} = 0$
l'arête est préservée par une chaîne de longueur 2

$c_{sg} = 0.69$
l'arête est préservée par une chaîne de longueur 3

FIG. 2.iv – Coûts de préservation des arêtes par des chaînes de différentes longueurs.

établissons : les points d'intérêt n'ont pas tous la même importance, les plus contrastés le sont plus que les autres. Dans ce niveau de coût rentre donc précisément en compte la valeur de contraste des points d'intérêt.

Niveau du point

Ceci signifie tout d'abord que les erreurs commises au niveau du point sont plus graves pour ceux qui sont le plus contrastés. Il semble alors juste de considérer que, lorsque l'on apparie deux sommets ensemble, la distance entre leurs masques soit pondérée par l'importance des sommets. Pour cela, on ordonne les masques par ordre de contraste décroissant et on utilise leur position p dans cette chaîne. Notons G_1 et G_2 les graphes que l'on représente sous forme de chaînes, m_1 et m_2 les masques que l'on apparie, et p_1 et p_2 leurs positions respectives dans la chaîne à laquelle ils appartiennent. Alors, par exemple pour la distance de Hamming, on obtient :

$$\frac{d_H(m_1, m_2) \times \frac{1+|G_1|-p_1}{|G_1|} + d_H(m_1, m_2) \times \frac{1+|G_2|-p_2}{|G_2|}}{2}$$

Soient les deux graphes dont les sommets ont été ordonnés de la figure 2.v. Alors on obtient les pondérations suivantes :

- apparier a et α : $d_H(a, \alpha) \times 1$;
- apparier d et γ : $d_H(d, \gamma) \times 0.65$;
- apparier h et α : $d_H(h, \alpha) \times 0.56$;
- apparier h et η : $d_H(h, \eta) \times 0.15$.

L'intégration de l'information de contraste se retrouve également pour les points non appariés. Une nouvelle fois, la position dans la liste ordonnée des points peut être prise en compte : le coût doit être plus important si le masque non apparié se situe parmi les points les plus contrastés, et inversement. On peut donc pondérer le coût de non appariement obtenu précédemment par un facteur w_m :

$$w_m = \frac{|G| - p + 1}{|G|}$$

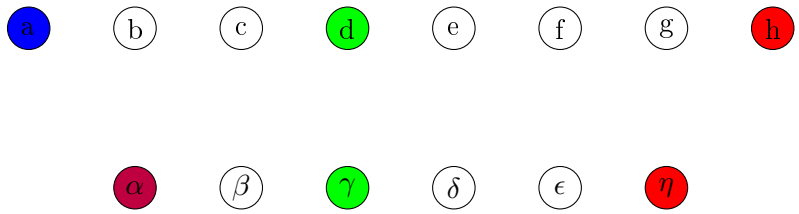


FIG. 2.v – Deux graphes dont on a ordonné les points d'intérêt : exemples de sommets appariés en fonction de leur position.

avec G le graphe auquel appartient le point non apparié de position p et de masque m. Ainsi, w_m est compris entre 1 (le point se situe en début de chaîne) et $\frac{1}{|G|}$ (le point se situe en fin de chaîne).

Ces pondérations peuvent évidemment être également d'un autre ordre (par exemple, inverse ou logarithmique), ou prendre en compte les différences entre les valeurs de contraste et non entre les positions des points dans la chaîne ordonnée.

Structure du graphe

Le raisonnement est identique en ce qui concerne la préservation des arêtes. Soit une arête entre deux points de position p_1 et p_2 dans G_1 (le raisonnement est le même pour G_2). Alors, si ces deux points sont appariés à deux point non adjacents de G_2, le coût de la structure du graphe c_{sg} est augmenté de :

$$c_{sg} += \frac{\frac{1+|G_1|-p_1}{|G_1|} + \frac{1+|G_1|-p_2}{|G_1|}}{2}$$

Des exemples de coûts d'arêtes sont représentés sur la figure 2.vi.

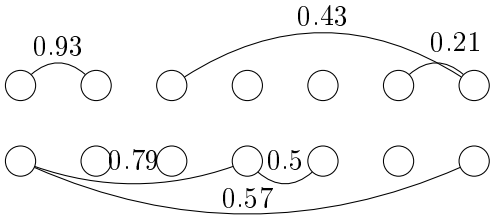


FIG. 2.vi – Coûts de non préservation des arêtes en fonction de leur position.

On pourrait pondérer de façon similaire le coût des arêtes préservées par des chaînes.

Le maintien de l'ordre de contraste

Rappelons-nous que nous partons du postulat que les points les plus contrastés sont les plus importants. Cela signifie également que si deux images font partie d'une même classe, alors les points d'intérêt les plus contrastés devraient caractériser les mêmes

zones. On considère donc qu'un bon appariement doit respecter l'ordre de contraste des points : points fortement contrastés appariés entre eux, et vice versa. En représentant, comme précédemment, les graphes par une chaîne de points ordonnés par ordre de contraste décroissant, cela revient à pénaliser tout croisement symboliquement créé par l'appariement.

Intéressons-nous à la distance de Kendall Tau [Kendall, 1970], qui, étant donné deux listes des mêmes éléments, évalue le nombre de paires n'étant pas dans le même ordre. Cette distance, pour deux listes τ_1 et τ_2, se définit comme suit :

$$K(\tau_1, \tau_2) = |(i,j) : i < j, (\tau_1(i) < \tau_1(j) \wedge \tau_2(i) > \tau_2(j)) \vee (\tau_1(i) > \tau_1(j) \wedge \tau_2(i) < \tau_2(j))|$$

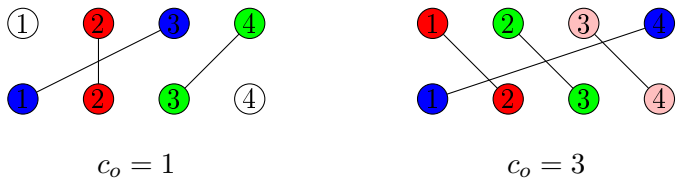
Par exemple, si $\tau_1 = \{3,1,4,5,2\}$ et $\tau_2 = \{1,4,5,3,2\}$ alors $K(\tau_1, \tau_2) = 3$ (les paires (1,3), (3,4) et (3,5) ne sont pas dans le même ordre dans τ_1 et τ_2).

La distance de Kendall Tau a ainsi une valeur de 0 si les deux listes sont identiques, et de $\frac{n \times (n-1)}{2}$ si elles sont en ordre inverse.

Il est possible d'adapter cette distance au cas de l'appariement des points ordonnés par ordre décroissant de contraste. Soit P l'ensemble des points du premier graphe étant appariés à Q, ensemble des points appariés du second graphe (on a donc $|P| = |Q|$). Pour tout point de rang $p \in P$, on note $a(p)$ le rang du point de Q qui lui est apparié. Alors, le coût c_o de l'appariement, au niveau du maintien de l'ordre, se définit comme suit :

$$c_o = |(p,q) \in P \times P : p < q, a(p) > a(q)|$$

Les exemples de la figure 2.vii présentent :
- un coût $c_0 = 1$ $(a(2) > a(3))$;
- un coût $c_0 = 3$ $(a(1) > a(4),\ a(2) > a(4),\ a(3) > a(4))$.



$c_o = 1$ $\qquad$ $c_o = 3$

FIG. 2.vii – Exemples de coûts de maintien de l'ordre non normalisés.

Cette adaptation de la distance de Kendall Tau, même si elle possède les mêmes bornes (comprise entre 0 et $\frac{n \times (n-1)}{2}$, pour les mêmes cas particuliers), ne compte pas la même chose : il s'agit ici de comptabiliser le nombre de croisements créés par l'appariement. Ainsi, sur l'exemple de Kendall Tau avec $K(\tau_1, \tau_2) = 3$, on a par contre $c_o = 5$, comme représenté sur la figure 2.viii.

Cependant, cette adaptation ne permet pas de prendre en compte l'étendue des croisements (S'agit-il d'un croisement créé par deux points proches ou très éloignés ?), ni la position des points concernés dans la chaîne ordonnée (S'agit-il d'un croisement

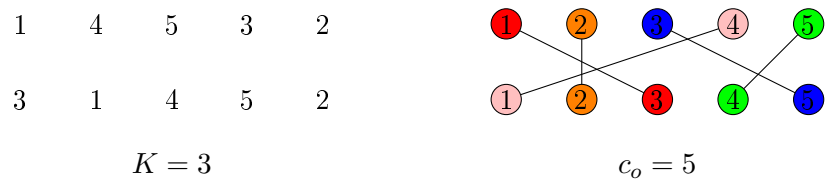


FIG. 2.viii – Comparaison de Kendall Tau et de son adaptation, c_o.

concernant des points très contrastés ou non ?). La figure 2.ix permet de ce fait de visualiser plusieurs croisements de même coût, $c_o = 1$.

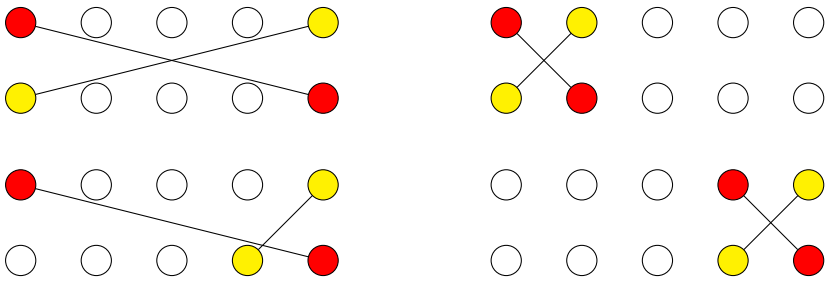


FIG. 2.ix – Plusieurs configurations avec $c_o = 1$.

Il est donc souhaitable de modifier le calcul de c_o, pour qu'il prenne en compte la position des points créant les croisements. Soient les points de rangs p_1 et $p_2 \in P$, respectivement appariés aux points de rang $a(p_1)$ et $a(p_2) \in Q$. Alors, le nouveau coût du croisement c_o vaut :

$$\frac{\frac{1+|P|-p_1}{|P|} + \frac{1+|Q|-a(p_1)}{|Q|} + \frac{1+|P|-p_2}{|P|} + \frac{1+|Q|-a(p_2)}{|Q|}}{4}$$

Des exemples de tels coûts de maintien de l'ordre peuvent être visualisés sur la figure 2.x. Ainsi, plus les croisements ont lieu au niveau de points fortement contrastés et plus leur coût est élevé. L'étendue du croisement est elle prise en compte non seulement par la position des points entrant en jeu, mais également de par le fait qu'un croisement très étendu en engendrera de nombreux autres dans la suite de l'appariement.

Il est bien sûr possible de mettre en place d'autres pondérations (inverse ou logarithmique, par exemple) en fonction de la position, ou d'utiliser les valeurs de contraste au lieu de la position de points.

2.2.4 Bilan sur la fonction de coût

Soient deux graphes G_1 et G_2 que l'on apparie. Intéressons-nous aux bornes des différents niveaux de coûts :

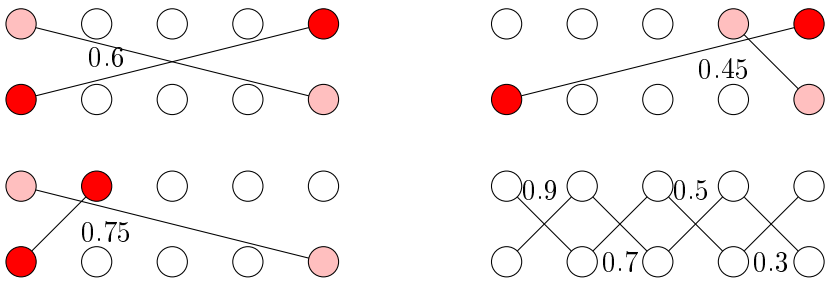


FIG. 2.x – Coûts de croisements en fonction de leur position.

- le niveau du point (pondéré ou non par le contraste) : clairement, qu'un point soit apparié ou non, le coût est compris entre 0 et 1. Le pire cas intervient quand aucun des points des deux graphes n'est apparié. Le coût total au niveau du point peut alors atteindre $|G_1|+|G_2|$ (borne n'étant jamais atteinte, puisque la distance aux masques moyens ne peut atteindre 1) ;
- la structure du graphe (pondérée ou non par le contraste) : nous nous intéressons à la version utilisant la fermeture transitive, qui est plus évoluée. son coût n'est théoriquement pas borné, étant donné que $\lim_{x\to+\infty} \ln(x) = +\infty$. Cependant, en pratique, dans le cas de graphes issus d'images, les arêtes sont toujours préservées par des chaînes relativement courtes. Notons que dans la triangulation de Delaunay, le nombre d'arêtes est strictement inférieur à 3 fois le nombre de sommets, ici il sera donc au total inférieur à $3 \times |G_1| + 3 \times |G_2|$.
- le maintien de l'ordre : chaque croisement a un coût compris entre 0 et 1. Dans le pire des cas, le nombre de croisement est de $\frac{min(|G_1|,|G_2|)\times(min(|G_1|,|G_2|)-1)}{2}$, ce qui correspond à la borne supérieure du coût total de maintien de l'ordre ;

Intéressons-nous aussi aux propriétés que la fonction de coût vérifie, étant donné deux graphes G_1 et G_2, leur appariement $A(G_1, G_2)$ et le coût associé $c(A(G_1, G_2))$:

- la non-négativité : clairement, $c(A(G_1, G_2)) \geq 0$. En effet, chacun des trois niveaux ne peut qu'augmenter le coût d'un appariement : tout est pénalité ;
- la symétrie : $c(A(G_1, G_2)) = c(A(G_2, G_1))$. Le niveau du point prend en compte tous les points appariés ou non des deux graphes, le maintien de l'ordre ne s'intéresse qu'à l'appariement, et la structure du graphe vérifie la préservation des arêtes pour les deux graphes ;
- l'unicité : $c(A(G_1, G_2)) = 0$ si et seulement si $A(G_1, G_2)$ est un appariement parfait entre deux graphes totalement identiques.

2.3 Analyse de la fonction de coût

Nous présentons dans cette partie les expérimentations menées pour analyser la fonction de coût. Il s'agira tout d'abord de tests portant sur le bruitage d'appariements,

puis de calculs de coûts d'appariements obtenus sur des bases de données d'images.

2.3.1 Bruitage d'appariements

Pour étudier le comportement de la fonction de coût, nous avons mis en place un protocole de bruitage d'appariements. Ainsi, en partant d'un appariement parfait entre les deux mêmes graphes, donc de coût 0, et en ajoutant du bruit, il est possible de visualiser la progression des différents niveaux de coût affectés par des modifications de l'appariement. Ceci permettra de visualiser dans quelle mesure chaque niveau entre en jeu dans le coût total. Dans l'idéal, la fonction de coût doit être résistante à un bruit modéré, puisqu'elle doit être valide pour des appariements imparfaits mais proches, c'est-à-dire que le coût ne doit pas augmenter de façon trop rapide.

Tout au long de cette partie, le coût au niveau du point correspond pour les points appariés à la distance de Hamming aux rotations près pondérée par la position des points ordonnés par contraste décroissant, pour les points non appariés il s'agit de la distance de Hamming aux masques moyens aux rotations près, pondérée également ; pour le respect de la structure nous utilisons le coût basé sur la fermeture transitive des graphes ; enfin, pour le maintien de l'ordre, il s'agit de l'adaptation de la distance de Kendall Tau pondérée.

Protocole

Le bruit, comme nous le verrons par la suite, peut intervenir à deux niveaux : au niveau des graphes eux-mêmes, ou au niveau de l'appariement. Pour chaque expérimentation, le protocole est le suivant :

- extraction de 250 points d'intérêt correspondant à la figure 2.xi ;
- triangulation de Delaunay des points, puis création de deux graphes identiques et d'un appariement parfait entre eux ;
- ajout de bruit comme décrit dans les paragraphes suivants, chaque bruit étant appliqué 250 fois (ce qui correspond au nombre de sommets de chacun des graphes) ;
- réitération de l'opération 20 fois ;
- calcul des 3 niveaux de coûts moyens pour chaque application de bruit sur les 20 itérations et création d'une courbe des résultats.

Modification des masques des points d'intérêt

Ce bruit consiste à choisir aléatoirement un sommet de chaque graphe dont le masque n'a pas encore été modifié, et à assigner à chacun un nouveau masque, choisi aléatoirement lui aussi. Lorsque l'on modifie les masques des points d'intérêt, on s'intéresse directement au niveau du point, et aux points appariés seulement (l'appariement de départ étant parfait, tous les points sont appariés). Les résultats peuvent être visualisés sur la figure 2.xii. Pour ces 250 masques modifiés, le coût atteint 45. Les autres niveaux de coût ne sont, en toute logique, pas atteints.

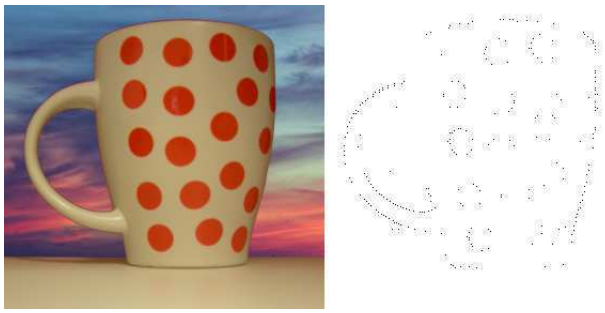


FIG. 2.xi – Image utilisée pour les expérimentations sur le bruit, et les points d'intérêt correspondant.

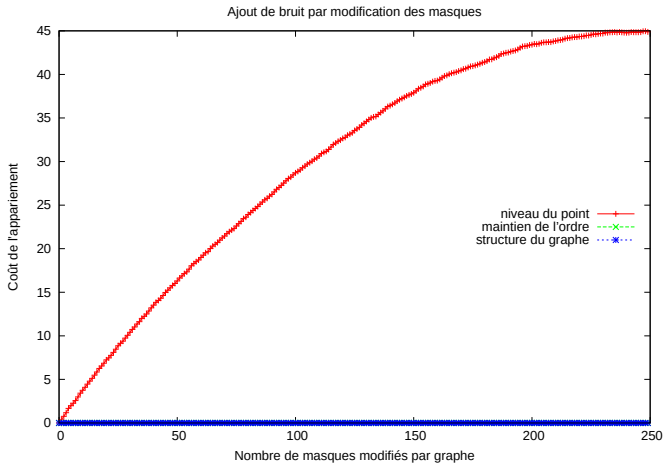


FIG. 2.xii – Ajout de bruit par modification des masques.

Désappariement de deux points

Il s'agit, à chaque itération, de tirer au hasard un sommet encore apparié, et de supprimer son appariement au sommet correspondant du second graphe. Comment le désappariement de deux points peut-il affecter la fonction de coût ? L'appariement de départ étant parfait, le désappariement provoque seulement une élévation du coût des points non appariés. Le comportement de la fonction de coût suite au désappariement répété de deux points peut être visualisé sur la figure 2.xiii : le coût atteint 50.

Modification de l'ordre de contraste des points d'intérêt

Dans cette expérimentation, on tire au sort à chaque itération deux sommets de chaque graphe, dont on échange la position dans la chaîne ordonnée des points par ordre de contraste décroissant. Lorsque l'on modifie l'ordre de contraste des points d'intérêt, on s'intéresse directement au niveau du maintien de l'ordre et donc à la création de

FIG. 2.xiii – Ajout de bruit par désappariement de sommets.

nouveaux croisements. Le comportement de ce niveau de coût peut être visualisé sur la figure 2.xiv : le coût se rapproche de 8000, ce qui est bien supérieur au niveau du point. Une pondération des coûts pourrait donc être envisagée.

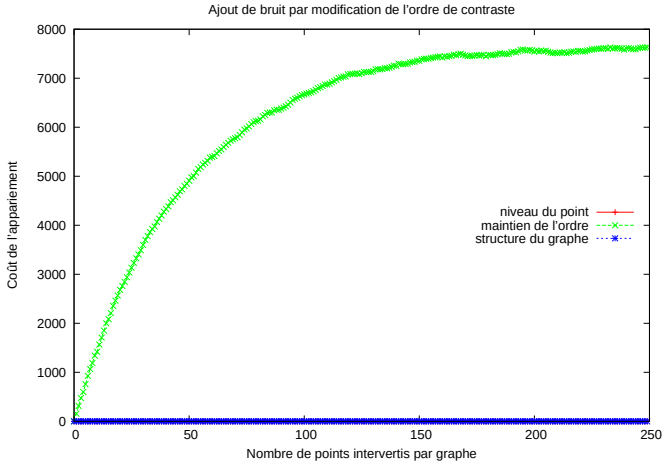


FIG. 2.xiv – Ajout de bruit par modification de l'ordre de contraste.

Réappariement de deux points

Dans cette expérimentation, nous choisissons aléatoirement deux sommets du premier graphe, puis récupérons et échangeons les sommets du second graphe auxquels ils étaient appariés. Que se passe-t-il lorsque l'on réapparie deux points ?

– au niveau du point : le coût des points appariés est soit augmenté (les masques

sont moins semblables qu'ils ne l'étaient), soit diminué (les masques sont plus semblables qu'auparavant), soit non modifié (pas de changement au niveau des masques). Le coût des points non appariés n'est pas modifié ;
- au niveau de la structure du graphe : on rappelle qu'une arête non préservée n'est coûteuse que si ses deux sommets sont appariés à deux points reliés par une chaîne de longueur minimale 3 (version avec la fermeture transitive). Encore une fois, le coût peut être augmenté (arête n'étant plus préservée), être diminué (longueur de la chaîne raccourcie) ou ne pas être modifié (compensation des longueurs) ;
- au niveau du maintien de l'ordre : il est soit augmenté (on a créé un ou plusieurs croisement), soit diminué (on a supprimé un ou plusieurs croisements), soit non modifié (les croisements se compensent).

Le comportement de la fonction de coût suite au réappariement répété de deux points peut être visualisé sur la figure 2.xv. Les coûts de maintien de l'ordre et de la structure du graphe se rapprochent respectivement de 4000 et 3000, tandis que celui du niveau du point reste très faible en comparaison.

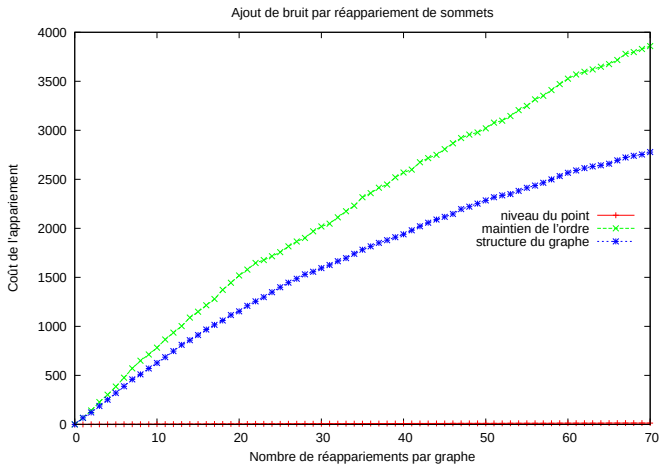


FIG. 2.xv – Ajout de bruit par réappariement de sommets.

2.3.2 Algorithmes GraphM et bases d'images COIL-100 et SIMPLIcity

Cette partie présente des expérimentations réalisées sur des bases d'images, dans le but d'obtenir des appariements et de calculer leurs coûts. Deux images d'une même classe devraient avoir un coût d'appariement plus faible que si elles appartenaient à deux classes différentes.

Deux bases d'images ont été utilisées. Il s'agit d'une part de COIL-1OO, qui se compose de 100 classes d'objets [Nene et al., 1996]. Chacun d'entre eux a été photographié en gros plan, par une caméra couleur, sur fond noir. Pour chaque objet, 72 poses, ayant entre elles 5 degrés de rotation, sont disponibles. Chaque image a une définition de

128 × 128 pixels. Pour un extrait, on peut consulter la figure 2.xvi. La seconde base utilisée est SIMPLIcity, qui contient dans sa version complète plus de 200000 images organisées en classes [Wang et al., 2001]. 10 classes sont disponibles librement et ont été utilisées pour les expérimentations. Pour chacune, 100 images de définition 384 × 256 sont accessibles, en format portrait ou paysage (se référer à la figure 2.xvii pour quelques extraits).

FIG. 2.xvi – Extraits de la base d'images COIL-1OO (classes 1, 12, 2 et 28).

Pour obtenir des appariements entre graphes, nous avons porté notre attention sur le package d'algorithmes GraphM[1], qui met à disposition un ensemble de méthodes fonctionnant avec des heuristiques et fournissant en sortie un appariement entre deux graphes d'entrée. Les algorithmes du package que nous avons utilisés retournent des appariements univoques, et approchés : ce sont des isomorphismes tolérants qui combinent le respect des arêtes et des étiquettes des sommets. Il s'agit de :

- I, algorithme d'identité (dans notre cas, l'appariement se fait donc dans l'ordre de contraste des points) ;
- U, algorithme d'Umeyama [Umeyama, 1988] basé sur les valeurs absolues des valeurs propres des matrices d'adjacence des graphes à apparier ;
- RANK [Singh et al., 2007] originalement utilisé pour l'alignement de protéines, cet algorithme fonctionne en deux étapes : la première associe un score pour chaque appariement possible de sommets , la seconde sélectionne le meilleur alignement global qui peut être localement sous optimal ;
- QCV et PATH [Zaslavskiy et al., 2008] tous deux basés sur une méthode de minimisation de fonctions convexes mais différant par leurs paramètres ;
- rand, appariement aléatoire.

Nous avons, pour débuter, testé uniquement deux classes d'images par base : obj1 et obj2 pour COIL-100, fleur et dinosaure pour SIMPLIcity. Le protocole des expérimentations, pour ces deux bases, est le suivant :

- extraction de 50 points d'intérêt par image via [Bres et Jolion, 1999] ;
- triangulation de Delaunay des points ;
- pour chaque couple d'images possible, application des 6 algorithmes d'appariement pré-cités. Pour cela, nous avons fourni des données d'entrée correspondant

[1] http://cbio.ensmp.fr/graphm/

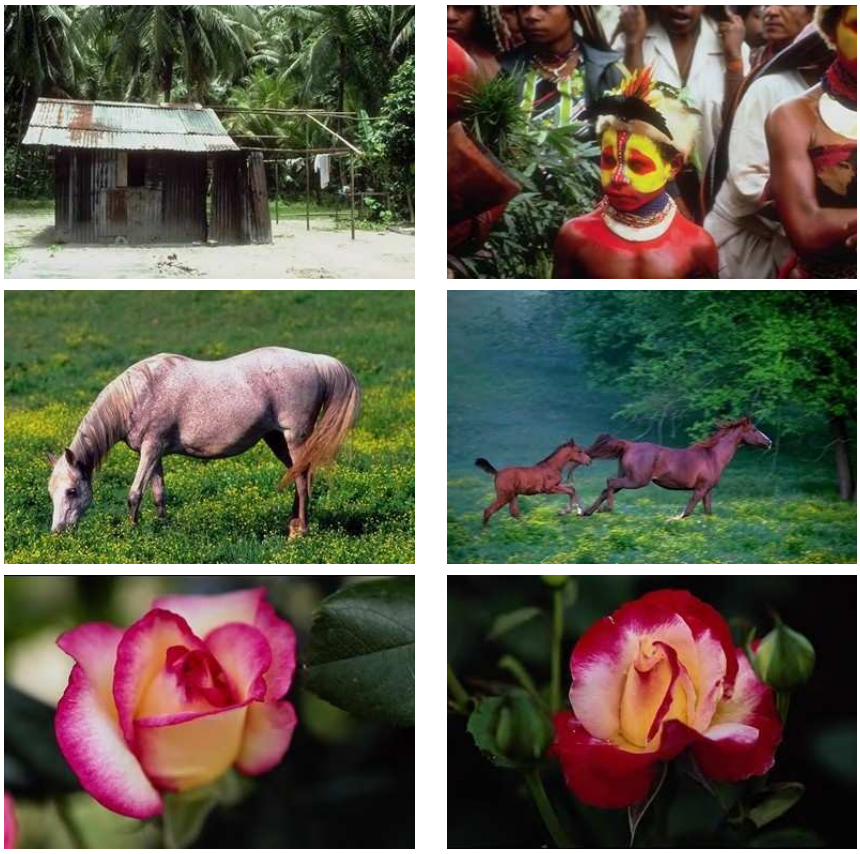


FIG. 2.xvii – Extraits de la base d'images SIMPLIcity (classes Afrique, cheval et fleur).

d'une part à la matrice d'adjacence de chaque graphe (ceci signifie qu'aucune arête n'a de poids particulier), et d'autre part aux coûts de substitution des sommets (dans le cas présent, nous avons utilisé le minimum de la distance de Hamming entre les masques, modérée par leur position dans l'ordre de contraste). Tous les points ont été appariés ;

– calcul des distances intra et inter-classes, avec le détail de chaque niveau de coût. Ici, le niveau du point correspond à la distance de Hamming en fonction de la position des points, celui du maintien de l'ordre à l'adaptation de la distance de Kendall Tau et celui de la structure du graphe à la pénalisation des arêtes non directement préservées (coût divisé par 2 pour une préservation par un chemin de longueur 2).

Les résultats concernant COIL-100 sont visibles dans le tableau 2.1, ceux de SIMPLIcity sur le tableau 2.2. Ces résultats ne sont pas pertinents : ils ne permettent pas de discerner de différence notable entre les distances inter et intra-classes. Le détail des niveaux de coût n'apporte pas d'information supplémentaire permettant de vérifier si l'un d'entre eux semble être discriminant.

	I	U	RANK	QCV	PATH	rand
obj1 : distance intra-classe						
minimum	285	531	527	391	311	587
niveau point	12	13	7	6	13	8
maintien ordre	0	223	233	295	258	331
structure graphe	274	296	287	90	41	248
maximum	454	771	790	759	551	778
niveau point	11	9	10	11	10	12
maintien ordre	0	401	361	368	400	331
structure graphe	443	362	418	379	141	435
moyenne	368	650	659	588	423	700
niveau point	10	10	10	10	10	10
maintien ordre	0	311	309	310	313	331
structure graphe	358	329	312	268	100	359
obj2 : distance intra-classe						
minimum	275	481	530	276	198	629
niveau point	8	7	10	5	5	10
maintien ordre	0	224	217	201	162	331
structure graphe	268	249	304	69	32	288
maximum	441	772	784	662	544	769
niveau point	9	10	10	9	12	11
maintien ordre	0	409	388	328	400	331
structure graphe	432	353	385	326	132	428
moyenne	359	629	654	517	388	702
niveau point	8	9	10	8	8	10
maintien ordre	0	307	312	286	295	331
structure graphe	351	313	332	223	85	361
distance inter-classes						
minimum	294	506	506	430	302	623
niveau point	9	10	10	14	10	11
maintien ordre	0	219	219	250	238	331
structure graphe	285	277	277	166	54	281
maximum	451	790	800	743	572	795
niveau point	12	11	12	11	11	11
maintien ordre	0	422	401	399	414	331
structure graphe	439	357	387	333	147	453
moyenne	370	650	661	589	428	703
niveau point	11	11	11	11	11	11
maintien ordre	0	314	312	315	311	331
structure graphe	359	325	338	262	105	361

TAB. 2.1 – Résultats sur deux classes de la base COIL-100.

	I	U	RANK	QCV	PATH	rand
fleur : distance intra-classe						
minimum	283	482	494	406	277	610
niveau point	10	10	9	11	11	11
maintien ordre	0	229	204	251	218	331
structure graphe	273	243	281	145	48	269
maximum	425	755	748	734	525	757
niveau point	9	11	13	11	16	11
maintien ordre	0	378	356	393	334	331
structure graphe	416	367	379	330	175	416
moyenne	357	634	647	573	414	691
niveau point	11	11	11	11	11	11
maintien ordre	0	309	310	311	311	331
structure graphe	347	314	326	252	93	349
dinosaure : distance intra-classe						
minimum	283	533	548	445	293	632
niveau point	12	10	11	11	11	10
maintien ordre	0	263	199	283	204	331
structure graphe	271	260	338	151	78	291
maximum	449	800	779	732	530	779
niveau point	11	13	11	12	12	11
maintien ordre	0	407	360	384	371	331
structure graphe	438	381	408	336	147	437
moyenne	369	648	664	586	424	701
niveau point	11	11	11	11	11	11
maintien ordre	0	312	315	311	311	331
structure graphe	358	325	338	263	101	358
distance inter-classes						
minimum	292	521	536	407	285	627
niveau point	15	16	15	11	14	14
maintien ordre	0	259	237	289	216	331
structure graphe	278	246	284	107	55	282
maximum	436	783	798	726	554	784
niveau point	14	15	15	14	15	13
maintien ordre	0	381	389	387	420	331
structure graphe	422	387	393	325	119	440
moyenne	368	646	658	583	423	699
niveau point	15	14	14	15	14	14
maintien ordre	0	312	312	312	311	331
structure graphe	353	319	332	257	98	354

TAB. 2.2 – Résultats sur deux classes de la base SIMPLIcity.

Pourquoi de tels résultats ? Tout d'abord, concernant SIMPLIcity, on peut noter que les classes sont parfois très hétérogènes, ce qui complique notre tâche : les graphes se ressemblent très peu pour certaines classes. Concernant COIL-100, même si les objets sont différents les uns des autres, le fait qu'ils soient tous mis en scène de la même façon rend sans doute l'approche par graphes mal adaptée. De plus, les rotations allant jusqu'à 360°, il serait illusoire de croire que les graphes n'en soient pas fortement modifiés à l'intérieur d'une même classe. Outre ces points, plusieurs choses peuvent être remises en cause : la façon d'extraire les graphes, notre fonction de coût, et les algorithmes utilisés. Il est difficile de savoir d'où provient réellement le problème : ainsi nos coûts ne sont peut-être pas assez discriminants (le niveau du point, notamment, est assez pauvre en information, et nous pourrions utiliser des étiquettes plus riches : les travaux ayant de bons résultats sur ces bases d'images se basent généralement sur des informations de couleur). Dans notre approche, nous avons souhaité rester plus proche de la théorie des graphes, prenant en compte leur structure avant tout, et ne considérant les labels que comme un enrichissement. Or, il semblerait que dans le cas de l'image, ces enrichissements, loin d'être accessoires, soient primordiaux, bien que nuisant à la généralité des méthodes.

2.3.3 Algorithme SoftAssign et images de maisons de Hancock

Pour aller plus loin, nous avons mené de nouvelles expérimentations, avec un autre algorithme et une autre base d'images. Il s'agit d'une part de l'algorithme SoftAssign [Gold et Rangarajan, 1996], qui fonctionne avec l'heuristique dite d'assignement gradué. Il s'agit de satisfaire à la contrainte d'un appariement univoque, en évitant de tomber dans un minimum local, le tout combiné à une méthode visant à accroître l'efficacité. Cet algorithme a déjà été utilisé et adapté dans le cadre de l'appariement de surfaces protéiques [Lozano et Escolano, 2006]. Pour les images, nous avons porté notre choix sur une base ayant déjà donné lieu à des travaux sur des représentations sous forme de graphes de Delaunay et sur des méthodes d'appariements [Luo et al., 2003]. Il s'agit de trois classes d'images en niveaux de gris, de définition 159 × 159 pixels, représentant des maisons, parfois entourées d'objets, soumises à de faibles rotations. Chaque classe comporte 10 images, dont les points d'intérêt ont déjà été extraits (voir la figure 2.xviii pour des exemples de telles images).

Le protocole des expérimentations est le suivant :

- utilisation des points d'intérêt fournis avec chacune des trois classes ;
- triangulation de Delaunay des points ;
- pour la première image de chaque classe, application de l'algorithme SoftAssign avec les 9 autres images de sa classe. Nous avons pour cela fourni un coût de substitution des sommets correspondant au minimum de la distance de Hamming entre les masques pondérée par leur position, et comme coût de substitution des arêtes la matrice d'adjacence de la triangulation de Delaunay ;
- calcul des coûts des appariements. Ici, le niveau du point correspond à la distance de Hamming en fonction de la position des points, celui du maintien de l'ordre à l'adaptation de la distance de Kendall Tau et celui de la structure du graphe à la

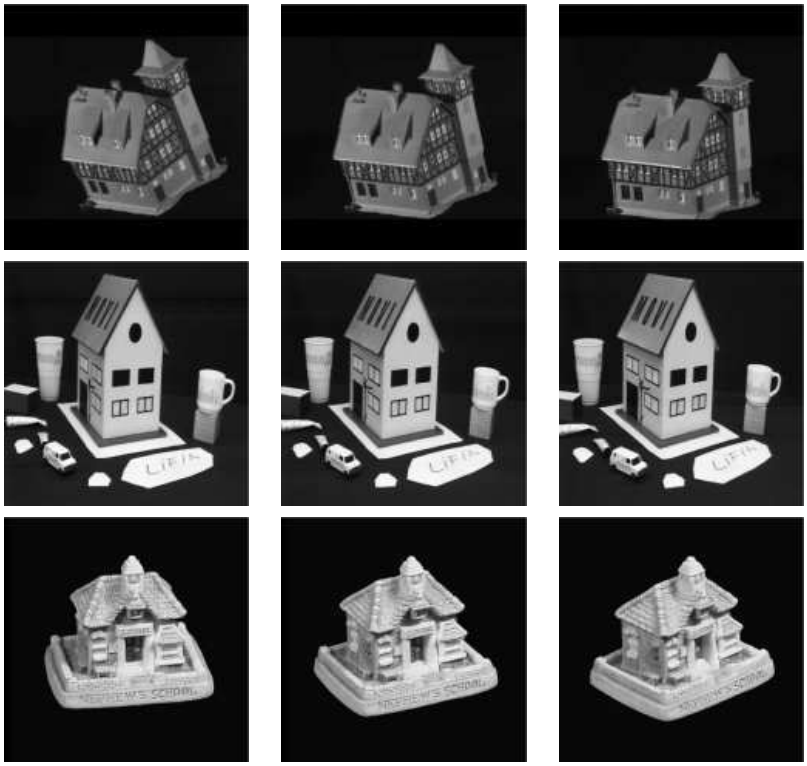
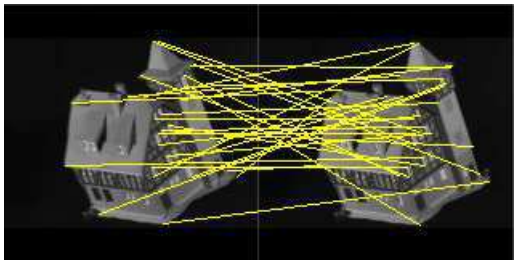


FIG. 2.xviii – Extraits d'images des classes cmu, movi et york.

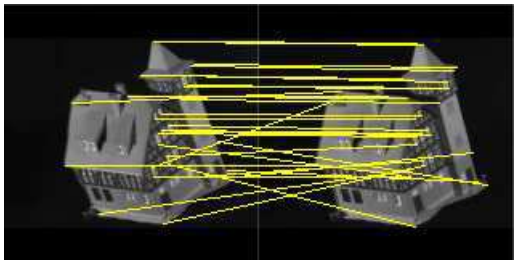
version utilisant la fermeture transitive des graphes.

Les tableaux 2.3, 2.4 et 2.5 présentent les résultats obtenus sur la classe cmu (la classe movi donne des résultats similaires mais difficilement visualisables à l'œil car le nombre de points est plus important, la classe york fournit quant à elle des résultats de moins bonne qualité). Ces résultats ont été arrondis à l'entier le plus proche. Mis à part deux appariements, la majorité est visuellement correcte, ce qui est un point positif : SoftAssign semble être moins dépendant des étiquette assignées aux graphes, et attacher une plus grande importance à l'obtention d'un appariement se rapprochant le plus possible d'un isomorphisme. Les informations des différents niveaux de coûts sont importantes : on se rend compte une nouvelle fois, comme pour COIL-100 et SIMPLIcity, que le niveau du point et le maintien de l'ordre ne sont pas discriminants. Par contre, la structure du graphe l'est : son coût est largement plus faible quand l'appariement est bon. La version utilisant la fermeture transitive des graphes semble donc plus probante. De plus, on peut aussi en déduire que les arêtes semblent effectivement apporter une information discriminante.



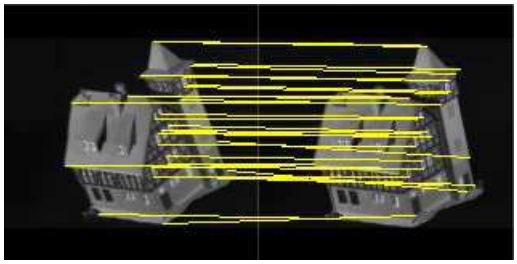
points appariés : 9
points non appariés : 0
maintien de l'ordre : 94
structure du graphe : 121
total : 223

cmu 1 cmu 2



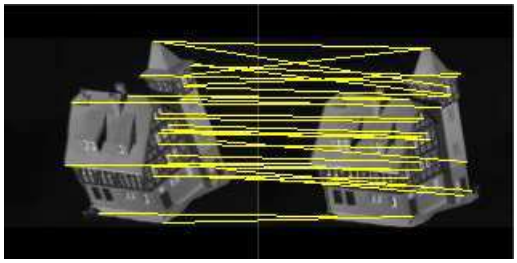
points appariés : 8
points non appariés : 0
maintien de l'ordre : 70
structure du graphe : 1
total : 79

cmu 1 cmu 3



points appariés : 7
points non appariés : 0
maintien de l'ordre : 77
structure du graphe : 0
total : 84

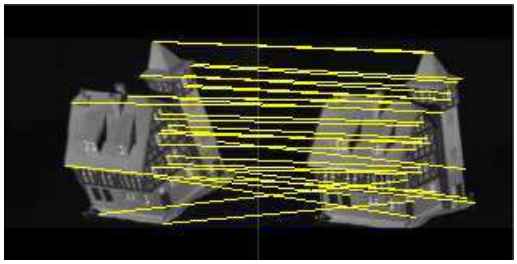
cmu 1 cmu 4



points appariés : 8
points non appariés : 0
maintien de l'ordre : 88
structure du graphe : 1
total : 97

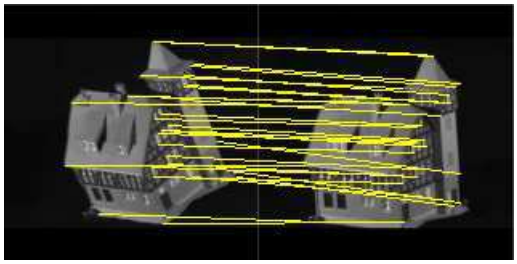
cmu 1 cmu 5

TAB. 2.3 – Appariements sur la classe cmu (1/3).



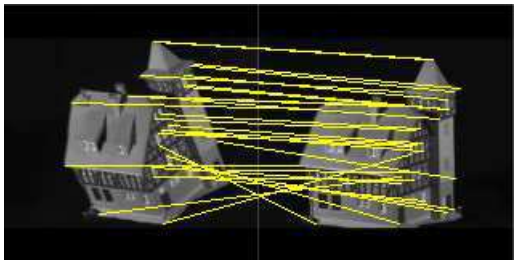
points appariés : 7
points non appariés : 0
maintien de l'ordre : 94
structure du graphe : 0
total : 101

cmu 1 cmu 6



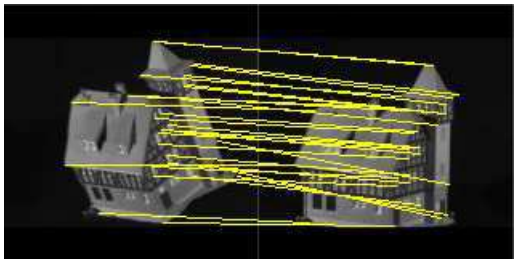
points appariés : 6
points non appariés : 0
maintien de l'ordre : 104
structure du graphe : 0
total : 111

cmu 1 cmu 7



points appariés : 7
points non appariés : 0
maintien de l'ordre : 87
structure du graphe : 15
total : 110

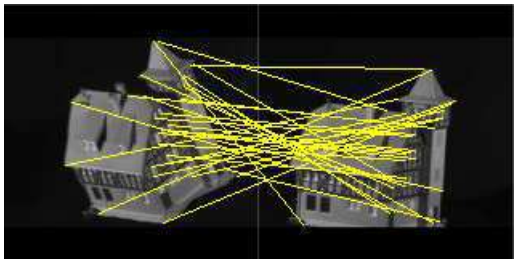
cmu 1 cmu 8



points appariés : 8
points non appariés : 0
maintien de l'ordre : 106
structure du graphe : 0
total : 114

cmu 1 cmu 9

Tab. 2.4 – Appariements sur la classe cmu (2/3).

 cmu 1 cmu 10	points appariés : 7 points non appariés : 0 maintien de l'ordre : 103 structure du graphe : 46 total : 157

TAB. 2.5 – Appariements sur la classe cmu (3/3).

2.4 Modifications envisageables

Dans cette partie, nous analysons et critiquons les critères que nous avons utilisés.

2.4.1 Réévaluation du niveau du point

Les expérimentations ont mis en avant le fait que ce niveau de coût était, de par sa borne supérieure, négligeable par rapport aux deux autres. Mais il semble également qu'en ne considérant que ce niveau de coût, on ne parvienne pas à distinguer si l'appariement est bon ou non.

Clustering de points d'intérêt

Une première modification a consisté à remettre en cause l'assignation d'un sommet du graphe par point d'intérêt. En effet, il est possible que le détecteur de points d'intérêt reste sensible à un certain bruit. Or, l'existence de sommets supplémentaires ou en moins peut modifier assez fortement le graphe de Delaunay correspondant, ce qui compromet dès le départ l'appariement futur. De plus, les points d'intérêt semblent souvent définir ce qu'on peut appeler des zones d'intérêt, où ils sont le plus souvent concentrés. Ces deux considérations nous ont amenés à expérimenter le clustering de points d'intérêt, dont on peut visualiser un exemple sur la figure 2.xix. Chaque cluster est alors représenté par un centroïde, obtenu par une moyenne pondérée par le contraste des coordonnées des points le composant, ce qui permet de donner plus de poids aux points les plus contrastés et de limiter l'importance des points correspondant à du bruit. Un centroïde, qui est un sommet du graphe, est alors étiqueté par l'histogramme de ses masques et par la somme des contrastes de ses points.

Des expérimentations avec l'utilisation de clusters de points d'intérêt ont également été menées sur des classes de COIL-100. Le coût au niveau des clusters appariés a été basé sur la différence entre les histogrammes de masques des clusters. Ces expérimentations n'ont pas permis d'obtenir des résultats significativement meilleurs que précédemment.

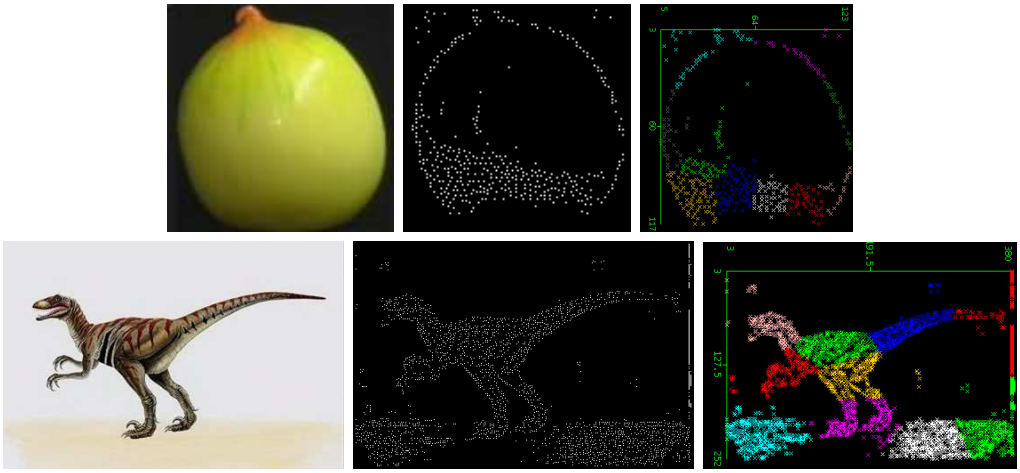


FIG. 2.xix – Clustering de points d'intérêt.

Distance d'édition de masques

Nous avons également souhaité vérifier la robustesse de l'extraction des points d'intérêt en ce qui concerne leur position spatiale dans l'image. En effet, la façon dont nous calculons le coût au niveau du point part du principe que les points d'intérêt que, à l'œil, nous voudrions apparier, ont des masques très semblables. Deux exemples de masques que l'on voudrait voir appariés peuvent être visualisés sur la figure 2.xx. On remarque clairement que les points que nous désirons associer l'un à l'autre n'ont pas le même masque : l'extraction des points d'intérêt est robuste, mais pas au pixel près. Un décalage d'un pixel sur le centre du masque peut engendrer un masque très différent, alors que ce sont pourtant bien ces deux points qu'il faut apparier. C'est peut-être bien cet aspect qui est déterminant et non pris en compte par notre fonction de coût.

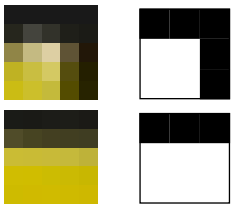


FIG. 2.xx – Deux voisinages de points que l'on souhaiterait associer et les masques 3 × 3 correspondant.

Pour cela, une piste que nous souhaitons étudier est celle de la distance d'édition entre masques. Rappelons que la distance d'édition, introduite sur les chaînes par Levenshtein [Levenshtein, 1966], se définit comme la plus petite (ou la moins coûteuse) suite d'opérations permettant de passer d'une chaîne à l'autre. Les trois opérations possibles

sur les lettres de l'alphabet qui constituent la chaîne sont la suppression, l'insertion ou la substitution d'une lettre. En ce qui concerne les points d'intérêt, la non conservation au pixel près nous mène à considérer une opération d'édition spéciale, la translation d'un masque. En effet, prenons en compte deux points que l'on apparie, mais dont les masques diffèrent à cause d'un léger décalage dans la détection du point d'intérêt. Or, si l'on translate les masques dans les images dont ils sont respectivement issus, on doit pouvoir retrouver les deux masques très semblables qui correspondent. Il s'agit donc de calculer une nouvelle distance entre masques, en trois temps (voir la figure 2.xxi) :

- translation du premier masque dans son image ;
- transposition de ce masque dans la seconde image ;
- translation de la position où le masque a été transposé à la position initiale du second masque.

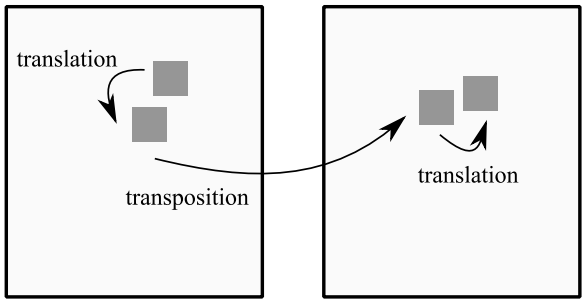


FIG. 2.xxi – Principe de la distance d'édition entre masques.

Le calcul de cette distance d'édition fera donc intervenir deux matrices de coûts de translation de masques, une pour chaque image. Ces matrices donneront le coût de passage du masque initial à un autre masque de la même image, par translation (on peut pour cela utiliser la distance de Manhattan ou la distance euclidienne). La phase de transposition, quant à elle, peut faire intervenir une opération de substitution des pixels d'un masque (on peut pour cela utiliser la distance de Hamming). Une autre solution est de considérer une distance d'édition entre chaînes de deux dimensions (ici, les masques), comme c'est le cas de certains travaux [Baeza-Yates, 1998].

Taille des masques et étiquetage

Enfin, outre ces recherches sur le clustering de points d'intérêt ou leur distance d'édition, nous pouvons encore envisager d'augmenter la taille des masques, qui pour l'instant a toujours été de 3×3 pixels. En effet, comme on peut se demander si l'étiquetage d'un point par son seul masque est assez significatif ou discriminant, la prise en compte d'un voisinage plus grand pourrait apporter plus d'information. De plus, la taille des masques pourrait être variable à l'intérieur d'une même image, en fonction du point d'intérêt considéré (par exemple, si la zone contient beaucoup de points d'intérêt, ils pourraient être résumés par un masque de taille importante). De plus, nous pourrions

	1	2	3	4	5	6	7	8
obj1, 0	 41,119	 47,42	 43,23	 51,28	 45,71	 44,119	 47,83	 51,121
obj1, 5	 43,117	 55,33	 43,57	 49,82	 42,19	 51,74	 46,47	 55,29
obj1, 10	 49,34	 41,25	 43,78	 43,76	 49,24	 43,61	 41,19	 47,59

TAB. 2.6 – Points d'intérêt de trois images par ordre de contraste : voisinage et coordonnées.

enrichir les labels des sommets, avec un descripteur tel que SIFT, plus usité de nos jours (voir la partie 1.1.2).

2.4.2 Prise en compte de l'information de contraste

Tout comme celui du point, le coût de maintien de l'ordre ne donne pas des résultats probants lors des expérimentations d'appariement que nous avons menées. On peut se demander également si la pondération des niveaux de coût par l'information de contraste est pertinente. Ces interrogations nous ont conduit à remettre en cause deux choses : la définition du contraste et la façon dont nous l'interprétons, mais aussi la méthode utilisée pour calculer le coût de maintien de l'ordre.

Retour sur la notion de contraste

Nous avons tout d'abord souhaité vérifier si les points d'intérêt ordonnés par contraste décroissant correspondaient aux zones de l'image que l'on souhaiterait, visuellement, apparier. Le tableau 2.6 présente les premiers points d'intérêt de trois images consécutives de la base COIL-100, par ordre de contraste décroissant. Il permet de visualiser leur voisinage et leurs coordonnées. Dans cette base et dans ce cas précis d'images consécutives, les coordonnées des points changent peu, deux points aux coordonnées proches sont donc très probablement appariés par un appariement parfait. Or, on remarque dès le premier point que celui de l'objet ayant eu un rotation de 10° (troisième ligne) a des coordonnées assez éloignées (rappelons que ces images ont une définition de 128×128 pixels) de celles du premier point des deux autres objets. Des différences sont également visibles pour les autres points. Partir du postulat que l'ordre de contraste doit être fortement pénalisé semble donc peu judicieux. Peut-être faudrait-il plutôt utiliser la valeur de contraste, et non l'ordre. Mais ceci reste à confirmer.

Le contraste, rappelons-le, se définit comme une mesure de perception de la différence de luminosité entre deux zones de couleur. Plus sa valeur est élevée et plus grande est la

FIG. 2.xxii – Capture d'écran de l'application sur la notion de contraste fournie aux étudiants.

différence. Parmi les mesures les plus courantes, on peut citer (voir la partie 1.1.1 pour plus de détails) : le contraste "simple", le contraste de Weber, le contraste de Michelson, et le contraste RMS. Pour comparer ces mesures, nous avons choisi, à la main, quelques points d'intérêt sur des images, et avons comparé l'ordre de contraste ainsi obtenu (en prenant un voisinage V_8). Les mesures fournissent des ordres relativement similaires, et il est difficile de détecter une mesure qui serait plus adaptée qu'une autre au cas des vignettes.

Nous avons cependant souhaité mesurer la cohérence entre ces définitions et la notion de contraste que peut ressentir une personne. Nous avons donc mis au point une expérimentation, en demandant à des étudiants de sélectionner, sur une image vignette ayant été agrandie, les 5 pixels qui leur semblaient être les plus contrastés, sans notion d'ordre (voir la figure 2.xxii pour une capture d'écran de l'application). Aucune définition précise ne leur a été fournie, la seule indication étant que par contraste, on entendait une opposition entre deux choses.

Nous avons ensuite récupéré les points sélectionnés et les avons dessinés sur les images dont ils provenaient (voir la figure 2.xxiii pour une illustration). Les résultats, visuellement, étaient très disparates, et il était rare que certains points aient été sélectionnés par plus d'un étudiant. La notion de contraste et son mode de calcul restent donc flous et il est difficile de s'assurer qu'elle doive être prise en compte par la fonction de coût. Contrairement à notre approche, le contraste est peut-être plus significatif lorsque l'on considère une zone, plutôt qu'un masque 3×3.

FIG. 2.xxiii – Vignettes (légèrement agrandies) et points d'intérêt extraits par les étudiants.

rang G_1	rang G_2	d_i	d_i^2
1	4	3	9
2	2	0	0
3	1	2	4
4	5	1	1
5	3	2	4

TAB. 2.7 – Corrélation d'un appariement

Coefficients de corrélation de rangs

Outre la notion de contraste en elle-même, le coût mis en place est lui aussi à critiquer. En effet, il atteint des valeurs très élevées, même si peu d'erreurs sont commises lors de l'appariement : une seule erreur peut engendrer $min(|G_1|, |G_2|) - 1$ croisements. Or, on peut considérer que le maintien de l'ordre doit être une notion plus globale qui ne pénalise pas directement chaque croisement, mais qui juge de la cohérence globale de l'appariement par rapport aux données de contraste possédées. On peut ainsi s'intéresser à des coefficients de corrélation de rangs.

Le coefficient de corrélation de Spearman ρ, de valeur comprise entre -1 (corrélation inverse) et 1 (totale corrélation) s'intéresse aux rangs pris par deux variables :

$$\rho = 1 - \frac{6 \sum d_i^2}{n(n^2 - 1)}$$

avec d_i la différence entre les rangs des valeurs correspondantes. Ainsi, si l'on reprend l'exemple de la figure 2.viii, on obtient les données du tableau 2.7. Alors, $\rho = 1 - \frac{6 \times 18}{5(25-1}) = 0.1$, ce qui indique une très faible corrélation et juge donc d'une mauvaise qualité au niveau du maintien de l'ordre. La façon d'intégrer un coefficient de corrélation de rangs à la fonction de coût pour le maintien de l'ordre est à étudier.

2.4.3 Apparier deux graphes de taille différente

Nous avons déjà indiqué que, pour éviter que l'appariement de meilleur coût ne soit celui où aucun point n'est apparié, il est nécessaire de pénaliser également les points non appariés. Ceci se justifie également si l'on apparie deux graphes de taille différente. Or, seul le niveau du point en est affecté. Comment pouvons-nous intégrer les points non appariés au maintien de l'ordre et à la structure du graphe ?

Maintien de l'ordre

Concernant le maintien de l'ordre, la question est la suivante : soient deux graphes $G_1 = (V_1, E_1)$ et $G_2 = (V_2, E_2)$ que l'on a appariés, ayant respectivement $|V_1|$ et $|V_2|$ sommets, avec $|G_1| > |G_2|$. Combien créerait-on en moyenne de croisements si l'on ajoutait, après l'appariement initial, $|V_1| - |V_2|$ sommets au second graphe ? Plusieurs solutions sont envisageables. On peut tout d'abord considérer que les sommets fictifs seraient moins importants que les sommets pré-existants, puisqu'ils n'ont pas été extraits par le détecteur de points d'intérêt. Dans ce cas, leur contraste étant inférieur, ils seraient donc ajoutés en fin de la chaîne des points ordonnés par ordre de contraste décroissant. Une autre possibilité est d'estimer que les points ajoutés étaient manquants lors de l'appariement initial, et qu'ils auraient donc dû se trouver à la bonne place dans la chaîne. Dans ce cas, ils sont insérés à la place des non appariés correspondants. Enfin, on peut tenir compte du fait qu'en moyenne, dans une liste de n éléments, le nombre d'inversions est de $\frac{n(n-1)}{4}$. On pourrait donc ajouter d'une façon ou d'une autre au coût initial du maintien de l'ordre ce nombre moyen de croisements qui seraient créés par les $n = |V_1| - |V_2|$ points.

Structure du graphe

La prise en compte de points supplémentaires dans un des deux graphes pour calculer le coût de la structure du graphe est plus délicate, puisque l'ajout de points nécessite l'introduction de coordonnées - le coût étant directement basé sur la conservation des chemins dans la triangulation de Delaunay. Aucune propriété sur la longueur moyenne du chemin (en nombre d'arêtes) entre deux points dans la triangulation de Delaunay ne semble exister.

2.5 Remise en cause de l'approche

Nous sommes en mesure de reconnaître que l'approche que nous avons mise en place est sans doute trop naïve. Par exemple, le respect strict de la relation d'ordre sur les points d'intérêt, que nous avons liée à leur valeur de contraste, est largement discutable. Ainsi, nous ne sommes pas parvenus à mettre au jour une quelconque stabilité qui puisse être exploitée.

Le principal enseignement que nous en retirons est que, dans le cadre des alignements de graphes issus d'images, il ne s'agit sans doute pas de découvrir des bijections (ou quasi-bijections) entre les sommets. En effet, un graphe est certes riche en information,

mais c'est également une structure complexe, qui impose, pour que des appariements de tous les sommets aient un sens, que des images semblables aient un graphe semblable. Or, imaginons tout simplement que nous cherchions à reconnaître un monument, avec un fond représentant le ciel. Même en conservant exactement les mêmes paramètres de prise de vue, échelle, *etc.*, il suffit que le temps change (présence de nuages, par exemple) pour que de nouveaux points d'intérêt soient détectés. Ceci mène, une fois la triangulation de Delaunay réalisée, à deux graphes très différents. On peut penser que l'appariement en est d'ores et déjà fortement compromis. Les limites de notre approche sont ici très claires : en vérité, le nombre de points d'intérêt à apparier est sans doute très restreint. La logique est l'inverse de la nôtre : au lieu de chercher à tout apparier et pénaliser les erreurs de l'appariement, le bon choix serait plutôt de récompenser les quelques éléments justes détectables [Deng et al., 2006, Sidibé et al., 2007, Sidibe et al., 2008].

En tout état de cause, nous attaquer à ce problème intimement lié à l'image de la façon dont nous l'avions envisagé est difficile. Néanmoins, ces considérations ont influencé notre réflexion, et parmi les nombreux choix de travaux futurs possibles, nous avons décidé de restreindre le problème, afin de le recentrer sur l'isomorphisme de graphes. Nous verrons ainsi, dès le chapitre suivant (chapitre 3), que les problèmes de reconnaissance de formes que nous souhaitons résoudre sont analogues à certaines questions posées dans la théorie des graphes. Nous développerons ensuite, dans le chapitre 4, une méthode d'extraction de graphes à partir d'images. Nous verrons pourquoi nous jugeons ces graphes intéressants à la fois pour représenter des images, mais aussi pour répondre aux problématiques d'isomorphismes. Ainsi, dans le chapitre 5, nous tirerons parti de la planarité de ces graphes pour proposer des algorithmes d'isomorphismes efficaces.

3

Théorie des graphes au service de l'image

Sommaire

Nous avons montré dans le chapitre 1 qu'il était possible de représenter les images numériques sous forme structurée, et notamment sous forme de graphes. Par ailleurs, nous proposerons dans cette thèse une nouvelle méthode d'extraction de graphes plans à partir d'images, qui sera détaillée dans le chapitre 4. Nous avons ainsi fait le choix de ce modèle symbolique pour une double raison : en effet, les graphes ne se contentent pas d'être riches en information. Comme nous l'étudierons dans ce chapitre, plusieurs problématiques permettent de comparer deux graphes entre eux : il s'agit principalement de l'isomorphisme de graphes et de sous-graphes, mais aussi de distance d'édition entre graphes. Nous verrons que ces problématiques prennent tout leur sens lorsqu'il s'agit de comparer des graphes issus d'images. Nous commencerons ainsi par exposer les principales définitions liées aux graphes, avant de nous intéresser aux isomorphismes de graphes et à l'aide qu'ils peuvent apporter dans des domaines de reconnaissance de formes ou de classification d'images. La littérature dans ce domaine étant très fournie, nous ne pourrons aborder toutes les méthodes existantes (le lecteur pourra se référer notamment à [Bunke, 2000, Conte et al., 2004, Conte et al., 2007]).

3.1 Graphes et images

Débutons ce chapitre par les définitions usuelles liées à la théorie des graphes. Nous ne serons pas exhaustifs et nous limiterons aux concepts qui seront utiles tout au long de cette thèse.

3.1.1 Définitions et terminologie

Rappelons la définition d'un graphe non orienté introduite dans la section 1.3 :

Définition 3.1 (Graphe non orienté)
Un graphe non orienté G *est un couple* (V, E) *avec :*
- V *un ensemble fini non vide d'éléments appelés sommets ou nœuds (*vertices *en anglais) de* G *;*
- $E \subseteq V \times V$ *un ensemble fini d'arêtes (*edges *en anglais), qui sont des ensembles à deux sommets.*

Dans la même section, nous avions noté qu'il était fréquent d'assigner des labels ou étiquettes au sommets et/ou arêtes des graphes, pour les enrichir et obtenir ce que nous nommons un graphe étiqueté.

Définition 3.2 (Graphe étiqueté)
Soient L_V *et* L_E *deux ensembles d'étiquettes (respectivement pour les sommets et les arêtes de* G*). Un* graphe non orienté étiqueté G *est un quadruplet* (V, E, α, β) *avec :*
- V *un ensemble fini non vide d'éléments appelés sommets ou nœuds (*vertices *en anglais) de* G *;*
- $E \subseteq V \times V$ *un ensemble fini d'arêtes (*edges *en anglais), qui sont des ensembles à deux sommets ;*
- $\alpha : V \to L_V$ *la fonction d'étiquetage des sommets ;*
- $\beta : V \times V \to L_E$ *la fonction d'étiquetage des arêtes.*

Comment sont constitués de tels labels ? Les étiquettes des sommets sont des descripteurs des parties qu'ils représentent : s'ils indiquent des régions, on peut leur assigner des informations de couleur (moyenne, variance...), de texture, de taille, de forme ; s'ils sont des points d'intérêt, on leur assigne des caractéristiques (contraste, orientation des gradients, informations sur le voisinage...). Quant aux étiquettes des arêtes, qui représentent les relations entre les sommets, elles peuvent dénoter la longueur des frontières communes, la distance entre les sommets...

Pour compléter notre vocabulaire sur les graphes, ajoutons les éléments de terminologie suivants [Wilson, 1996] :
- la *taille* d'un graphe G est son nombre de sommets, noté $|V|$;
- un sommet $v \in V$ est *incident* à une arête $e \in E$ si v est une extrémité de e ;
- deux sommets sont *adjacents* s'ils sont incidents à une même arête ;
- le *degré* d'un sommet $v \in V$ est le nombre d'arêtes incidentes, noté $\delta(v)$;
- un sommet de degré 1 est appelé sommet *pendant* ;
- une arête incidente à un sommet de degré 1 est une arête *pendante* ;
- une *chaîne* $v_0 v_1 \dots v_l$ est une suite d'arêtes consécutives du graphe $(\{v_0 v_1\}, \{v_1, v_2\}, \dots, \{v_{l-1} v_l\})$;
- la *longueur* d'une chaîne est le nombre d'arêtes la composant ;
- un graphe G est *connexe* s'il existe une chaîne entre toute paire de sommets de G ;

– un *cycle* est une chaîne dont les sommets des extrémités sont identiques.

La figure 3.i nous permet d'illustrer ces différentes notions : le graphe non orienté qui y est représenté est étiqueté. Les labels des sommets sont des entiers de 1 à 6, ceux des arêtes sont des lettres de a à e. Ce graphe est de taille 6, car il a six sommets. Les sommets 1 et 2 sont incidents à l'arête a, ils sont donc tous deux adjacents. Le degré du sommet 1 est $\delta(1) = 3$: il possède 3 arêtes incidentes, qui sont a, b et c. La chaîne abd (ou $(\{2,1\},\{1,4\},\{4,5\})$) est de longueur 3. aec (ou $(\{2,1\},\{1,3\},\{3,2\})$) est un cycle de longueur 3. On remarque que ce graphe n'est pas connexe : le sommet 6 est isolé.

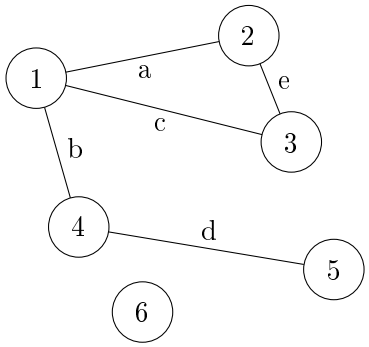


FIG. 3.i – Exemple de graphe non orienté étiqueté.

Un graphe peut également être décomposé en un ensemble d'éléments. De ce fait, on définit les notions de sous-graphe, graphes partiel et sous-graphe partiel.

Définition 3.3 (Sous-graphe (induit))
Le graphe $G' = (V', E')$ *est le* sous-graphe *de* $G = (V, E)$ *induit par* V' *si :*
– $V' \subset V$;
– *et* $E' = \{\{u, v\} \in E \mid u \in V',\ v \in V'\}$.

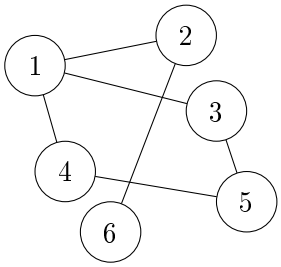
Ainsi, on obtient un sous-graphe en enlevant des sommets et toutes les arêtes incidentes à ceux-ci. Un sous-graphe (induit) de G peut donc être uniquement défini par un sous-ensemble de sommets de G, les arêtes étant une conséquence de ce choix.

Définition 3.4 (Graphe partiel)
Soit $G = (V, E)$ *un graphe. Alors, le graphe* $G' = (V, E')$ *est un* graphe partiel *de* G *si* $E' \subset E$.

On construit un graphe partiel en enlevant des arêtes au graphe d'origine, mais en conservant tous les sommets. Un graphe partiel peut se définir uniquement par ses arêtes.

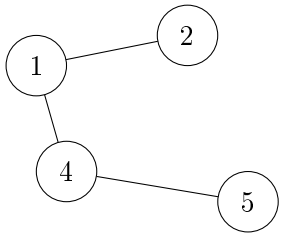
Définition 3.5 (Sous-graphe partiel)
Un sous-graphe partiel *est un graphe partiel d'un sous-graphe.*

Ces différents types de graphes obtenus par modification des sommets et/ou arêtes sont illustrés sur la figure 3.ii.



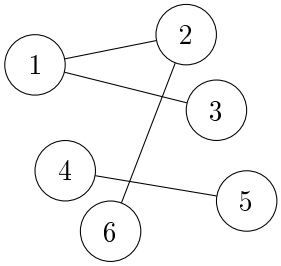
(a)

$G = (V, E)$
$V = \{1, 2, 3, 4, 5, 6\}$
$E = \{\{1, 2\}, \{1, 3\}, \{1, 4\}, \{2, 6\}, \{3, 5\}, \{4, 5\}\}$



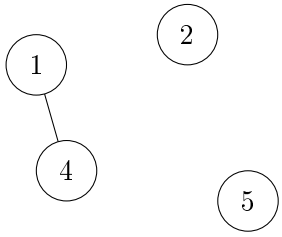
(b)

$G' = (V', E')$ sous-graphe induit de (a)
$V' = \{1, 2, 4, 5\}$
$E' = \{\{1, 2\}, \{1, 4\}, \{4, 5\}\}$



(c)

$G'' = (V, E'')$ graphe partiel de (a)
$E'' = \{\{1, 2\}, \{1, 3\}, \{2, 6\}, \{4, 5\}\}$



(d)

$G''' = (V', E''')$ sous-graphe partiel de (a) : graphe partiel de (b)
$E''' = \{\{1, 4\}\}$

FIG. 3.ii – Illustration des notions de graphe, sous-graphe, graphe partiel et sous-graphe partiel.

3.1.2 Images et planarité

Lorsque les graphes représentent des images, ils ont une propriété supplémentaire non négligeable, qui est celle de la planarité. En d'autres termes, leur représentation dans le plan est plane : on peut les dessiner de façon à ce que leurs arêtes ne se rencontrent jamais, sauf aux extrémités [Fusy, 2007].

Définition 3.6 (Représentation plane)
Une représentation plane *d'un graphe $G = (V, E)$ est un plongement η dans le plan tel que :*

- *η_V est une application injective liant les sommets de G à des points dans le plan ;*
- *η_E est une application liant les arêtes de G à des courbes du plan, telle que chaque extrémité d'arête $e \in E$ soit liée par η_V aux extrémités de la courbe $\eta_E(e)$;*
- *$\forall e, e' \in E$, les courbes $\eta_E(e)$ et $\eta_E(e')$ ne se rencontrent jamais, sauf aux extrémités qui leur sont communes.*

Définition 3.7 (Graphe planaire)
Un graphe G est planaire *s'il admet au moins une représentation plane.*

Définition 3.8 (Graphe plan)
Un graphe plan *est une représentation plane d'un graphe planaire.*

La figure 3.iii illustre le cas d'un graphe non planaire : il est impossible de le dessiner sans que ses arêtes ne se rencontrent. Par opposition, le graphe de la figure 3.iv, lui, est planaire. Sur cette illustration, trois plongements dans le plan possibles sont représentés : sur le premier, le graphe n'est pas plan (*i.e.* sa représentation n'est pas plane : deux arêtes se croisent). Les deux autres représentations sont planes, cependant elles diffèrent par une autre propriété : sur la deuxième, une arête est sous la forme d'une courbe alors que sur la troisième, toutes les arêtes sont des segments. Il a été montré [Fáry, 1948] que tout graphe planaire simple sans boucle peut être dessiné de façon à ce que toutes ses arêtes soient des segments.

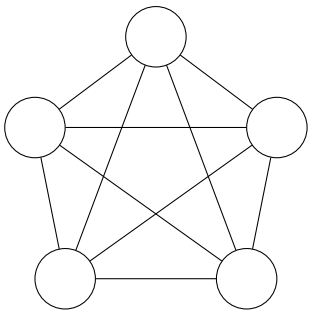


FIG. 3.iii – Un graphe non planaire.

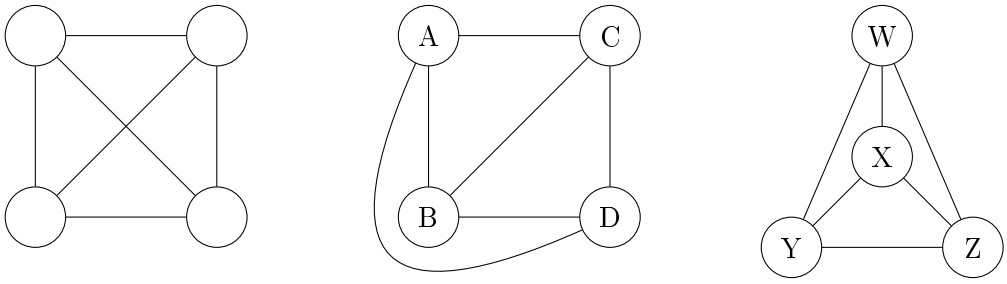


FIG. 3.iv – Trois représentations d'un même graphe planaire : une représentation non plane et deux représentations planes.

Dans un graphe plan, on peut définir un objet supplémentaire, qui sont les *faces*. Toute représentation plane d'un graphe divise l'espace en régions, appelées faces. L'ensemble des arêtes limitant la face, qui forme un cycle, est couramment appelé *frontière*. Ainsi, sur les deux représentations de droite de la figure 3.iv, on trouve respectivement les faces suivantes : ABC, BCD, ABD et ACD (qui est la face externe ou infinie) ; et WXY, WXZ, XYZ et WYZ (qui constitue également la face externe). De plus, la formule d'Euler indique que, quelque soit le plongement considéré d'un graphe planaire, le nombre de faces est constant et répond à une formule simple :

Théorème 3.1 (Formule d'Euler) *Soit η une représentation plane d'un graphe planaire connexe $G = (V, E)$, et soit $|F|$ son nombre de faces. Alors*

$$|V| - |E| + |F| = 2. \tag{3.1}$$

Ainsi, pour les deux graphes plans de droite de la figure 3.iv, on a $|V| = 4$, $|E| = 6$, $f = 4$ et $|V| - |E| + f = 4 - 6 + 4 = 2$. De plus, si $|V| \geq 3$, alors le corollaire suivant est vérifié :

Corollaire 3.1 (Nombre d'arêtes d'un graphe planaire)
Soit $G = (V, E)$ un graphe planaire connexe, avec $|V| \geq 3$. Alors

$$|E| \leq 3|V| - 6. \tag{3.2}$$

Une des conséquences de ce corollaire est que, lorsque l'on s'intéresse à la complexité d'algorithmes mettant en jeu des graphes planaires, il est possible de ne tenir compte que de $|V|$.

3.2 Problématiques d'isomorphismes et reconnaissance de formes

En reconnaissance de formes, ou dans d'autres domaines tels la classification supervisée ou non d'images, les buts, bien que divers, restent semblables. Il peut s'agir tout d'abord de décider si deux images représentent le/les même(s) objet(s), ou des objets très ressemblants. Si des graphes sont extraits de ces images, cela revient à se demander s'ils sont *isomorphes*. Savoir si une image est incluse dans une autre (par exemple, dans quelles images puis-je retrouver ce cheval ?) est une autre interrogation possible. Cette fois, cela correspond à retrouver un sous-graphe dans un graphe : c'est l'*isomorphisme de sous-graphe*. Enfin, on pourrait également vouloir déterminer quelles choses deux images ont en commun, et en particulier, quelle est la plus grosse partie commune. Il s'agit de la recherche de *plus grand sous-graphe commun*. Dans cette partie, nous allons nous intéresser tour à tour à ces trois problématiques.

3.2.1 Isomorphisme de graphes

Nous voici en possession de deux graphes issus d'images. Nous souhaitons savoir si ces images sont identiques : nous voulons donc comparer les deux graphes. Décider si deux graphes sont ou non structurellement identiques est une problématique appelée problème de l'isomorphisme de graphes.

Définition 3.9 (Isomorphisme de graphes)
Deux graphes $G_1 = (V_1, E_1)$ *et* $G_2 = (V_2, E_2)$ *sont* isomorphes *s'il existe une bijection* ϕ *de* V_1 *dans* V_2 *telle que*

$$\{i, j\} \in E_1 \Leftrightarrow \{\phi(i), \phi(j)\} \in E_2. \tag{3.3}$$

Cette définition signifie simplement que deux graphes sont isomorphes s'il existe une correspondance entre leurs deux ensembles de sommets, telle que toutes les arêtes soient respectées (voir par exemple la figure 3.v). L'isomorphisme de graphes peut ainsi être directement relié à l'appariement de graphes (voir la partie 2.1.1) : deux graphes sont isomorphes s'il existe entre eux un appariement bijectif, avec une contrainte dure sur les sommets et les arêtes.

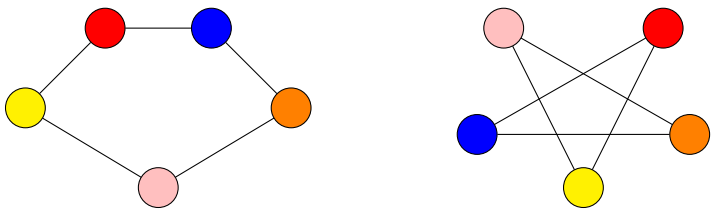


FIG. 3.v – Deux graphes isomorphes : l'appariement est dénoté par les couleurs.

Pour ce qui est de l'application aux images, prenons pour exemple l'illustration de la figure 3.vi[1] qui représente deux images, pour lesquelles les graphes ont été superposés (en noir). Il s'agit de graphes simples, des RAG, pour permettre une meilleure visualisation. Ces deux graphes sont parfaitement isomorphes : on peut trouver au moins un appariement bijectif entre eux (ici, représenté par des arcs bleus) qui respecte les sommets et les arêtes. Cet appariement, si on lui assignait un coût, serait parfait et donc de coût 0.

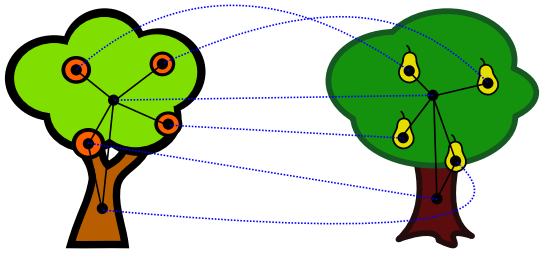


FIG. 3.vi – Deux graphes issus d'images (en noir) qui sont isomorphes : l'appariement des sommets est représenté par des arcs bleus.

Cela paraît tout à fait correct de conclure que les deux images sont identiques, ou tout du moins très proches. Pourtant, il est possible d'enrichir ces graphes, en leur assignant des labels : c'est le cas présenté par la figure 3.vii, où la couleur des régions a été assignée comme étiquette aux sommets. Même si les graphes sont toujours isomorphes, cette fois l'appariement n'est pas parfait, puisque les labels des sommets appariés ne sont pas les mêmes : le coût associé serait supérieur à 0. Il existe donc potentiellement plusieurs isomorphismes, certains étant *meilleurs* que d'autres.

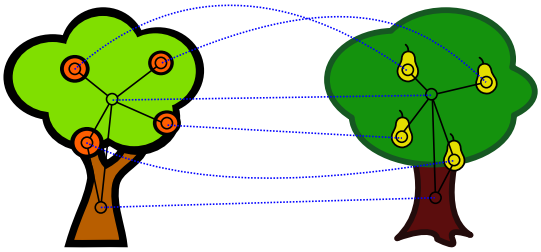


FIG. 3.vii – Deux graphes isomorphes, mais dont les labels des sommets diffèrent.

Qu'en est-il de la complexité du problème de l'isomorphisme de graphes ? Cela reste un problème ouvert [Fortin, 1996] :

[1] Les images de ce chapitre représentant des graphes issus d'images ont été réalisées en utilisant la librairie de *clipart* `http://www.openclipart.org/`, qui est de domaine public.

Complexité de l'isomorphisme de graphes Le problème d'isomorphisme de graphes dans le cas général appartient à la classe NP. Cependant, cela reste un problème ouvert : on ne sait pas s'il appartient à P ou s'il est NP-complet, ni même s'il appartient à co-NP. L'existence d'un isomorphisme entre deux graphes a ainsi sa propre classe de complexité : c'est un problème *isomorphique-complet*.

Cependant, même si aucun algorithme pour les graphes généraux ne fonctionne en temps polynomial dans le pire des cas, il est en pratique non trivial de trouver des graphes pour lesquels le problème de l'isomorphisme soit difficile à résoudre, même s'ils ont une taille importante [Fortin, 1996]. De plus, certaines classes de graphes particuliers peuvent avoir des complexités polynomiales. Néanmoins, la plupart de ces algorithmes polynomiaux dédiés à des classes de graphes particulières ont des constantes les rendant prohibitifs en pratique. D'autant plus qu'il est souvent non trivial de déterminer si un graphe appartient à une classe donnée.

Beaucoup d'approches de résolution d'isomorphismes se basent sur des invariants associés aux sommets, afin de réduire l'espace de recherche (pour un graphe de taille n, il y a en effet $n!$ appariements possibles). Un invariant est un nombre $i(v)$ lié à un sommet v (par exemple, le degré, invariant de base souvent utilisé pour les graphes non réguliers) tel que, s'il existe un isomorphisme qui apparie v et v', alors $i(v) = i(v')$ (notons que l'inverse ne tient pas : si deux sommets ont le même invariant, il n'existe pas forcément d'isomorphisme les appariant). Ainsi, l'algorithme *nauty* [McKay, 1981] propose plusieurs invariants : un nombre basé sur les sommets atteignables par une chaîne de longueur 2, un nombre basé sur le nombre de sommets à distance 1 à n de v... En fait, en pratique, les invariants sont souvent combinés les uns aux autres pour être plus discriminants. Leur temps de calcul peut alors devenir non négligeable dans le temps d'exécution de l'algorithme d'isomorphisme, le calcul de l'invariant idéal et performant pour un graphe donné étant, en outre, une tâche non triviale.

Une première méthode pour résoudre le problème de l'isomorphisme de deux graphes est de tenter d'en construire un incrémentalement, par essais-erreurs. À un petit sous-graphe du premier graphe, isomorphe à un du second, on ajoute peu à peu des sommets tout en conservant le sous-isomorphisme [Cordella et al., 1999, Cordella et al., 2001]. Les invariants sont alors utilisés pour sélectionner les sommets à ajouter et permettre de rejeter rapidement le plus possible de possibilités. Une autre approche consiste à calculer un identifiant unique par graphe. Deux graphes sont alors isomorphes s'ils ont le même identifiant. C'est également le cas de *nauty*, qui examine tous les automorphismes d'un graphe (qui sont des isomorphismes avec lui-même) et calcule un identifiant canonique. Mais il est également possible de penser l'isomorphisme de graphes comme étant un problème portant sur les matrices d'adjacences des graphes considérés. Une matrice d'adjacence est une des possibilités de structures de données représentant des graphes. Pour un graphe $G = (V, E)$, il s'agit d'une matrice booléenne A de taille $|V| \times |V|$ qui vérifie, $\forall i, j \in \{1, 2, \ldots, |V|\}$, $a_{ij} = 1$ si $\{i, j\} \in E$, et $a_{ij} = 0$ sinon. Certains algorithmes travaillent ainsi directement sur les matrices d'adjacence, se ramenant à un problème d'optimisation sur l'ensemble des permutations, en incluant parfois des notions de probabilité d'appariements de sommets, ou des noyaux [Gold et Rangarajan, 1996, Lozano et Escolano, 2004, Singh et al., 2007,

Zaslavskiy et al., 2008, Leordeanu et Hebert, 2009]. Une autre possibilité est de s'intéresser aux valeurs propres et vecteurs propres des matrices d'adjacence, invariants aux permutations [Umeyama, 1988, Shapiro et Brady, 1992, Caelli et Kosinov, 2004]. Ainsi, si deux graphes sont isomorphes, les valeurs et vecteurs propres seront les mêmes (mais l'inverse n'est pas forcément vrai). D'autres méthodes font elles appel à des arbres de recherche [Messmer et Bunke, 1999] utilisant des heuristiques pour élaguer l'espace de recherche, ou à une approche par programmation par contraintes [Sorlin et Solnon, 2004].

En fait, beaucoup de ces méthodes calculent un isomorphisme dit tolérant, c'est-à-dire que certaines contraintes dures sur les sommets et arêtes peuvent ne pas être respectées. En effet, ils est rare que deux images proches aient *exactement* le même graphe. Être tolérant permet une meilleure généralisation. Nous y reviendrons dans la partie 3.3, et verrons que ceci peut être conçu comme étant un calcul de distance entre graphes, pour lequel on souhaite minimiser une fonction de coût.

Revenons à des considérations de planarité. Il a été montré [Hopcroft et Wong, 1974] que pour la classe des graphes planaires, le problème de l'isomorphisme pouvait se résoudre en temps linéaire, cependant cet algorithme reste inexploitable en pratique, à cause d'une importante constante. Il fut par la suite amélioré en intégrant de la programmation parallèle [Jaja et Kosaraju, 1988], mais reste rédhibitoire. Nous avons déjà indiqué que dans le cas d'images, les graphes extraits étaient planaires. Mais ils sont surtout plans : leur plongement est donné. Dans ce cas, que penser par exemple des graphes de la figure 3.viii ? Ils sont bien isomorphes dans le sens de la définition, mais souhaitons-nous, dans le cas d'images, qu'ils le soient ?

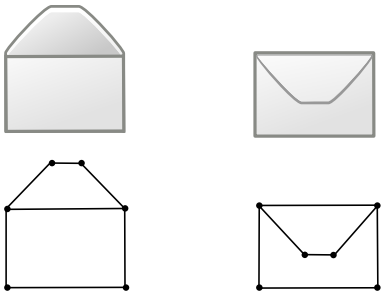


Fig. 3.viii – Deux graphes issus d'images isomorphes.

Ainsi, il existe une définition différente de l'isomorphisme, pour le cas plan. Nous y reviendrons en détails dans le chapitre 5 et proposerons un algorithme polynomial d'isomorphisme de graphes s'appliquant à ce cas particulier.

3.2.2 Isomorphisme de sous-graphe

On peut également vouloir savoir si un objet apparaît ou non dans certaines zones d'images. Cette fois, on cherche plutôt à retrouver un graphe (le motif que l'on recherche) dans un autre, plus grand (l'image où l'on recherche le motif). Décider si un graphe est ou non inclus dans un autre est appelé problème de l'isomorphisme de sous-graphe.

Définition 3.10 (Isomorphisme de sous-graphe)
Il existe un isomorphisme de sous-graphe *entre deux graphes* $G_1 = (V_1, E_1)$ *et* $G_2 = (V_2, E_2)$ *s'il existe une injection* ϕ *de* E_1 *dans* E_2 *telle que*

$$\{i, j\} \in E_1 \Leftrightarrow \{\phi(i), \phi(j)\} \in E_2. \tag{3.4}$$

Un isomorphisme de sous-graphe de G_1 dans G_2 est en fait un isomorphisme de G_1 dans un sous-graphe de G_2 : il existe une correspondance entre les sommets de G_1 et ceux de G_2, telle que toutes les arêtes de G_1 soient respectées (voir par exemple la figure 3.ix qui illustre un cas possible d'isomorphisme de sous-graphe). L'isomorphisme de sous-graphe peut ainsi être directement relié à l'appariement de graphes (voir la partie 2.1.1) : G_1 est isomorphe à un sous-graphe de G_2 s'il existe entre eux un appariement injectif, avec une contrainte dure sur les sommets et les arêtes si l'on considère un sous-graphe induit (ce qui est le cas ici) ; dure sur les sommets et souple sur les arêtes si l'on considère un sous-graphe partiel.

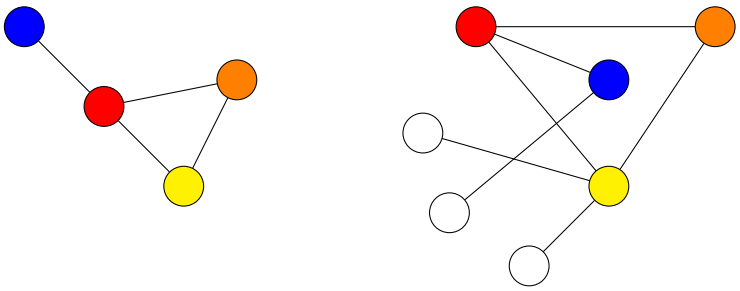


FIG. 3.ix – Isomorphisme de sous-graphe : l'appariement est dénoté par les couleurs.

Illustrons ces considérations dans le domaine de l'image, avec la figure 3.x qui représente deux images, pour lesquelles les graphes ont été superposés (en noir). À droite, il s'agit du motif que l'on cherche à retrouver dans l'image de gauche. Il existe bien un isomorphisme de sous-graphe entre eux : on peut trouver au moins un appariement injectif (ici, représenté par des arcs bleus) qui respecte les sommets et les arêtes. Cet appariement, si on lui assignait un coût, serait parfait et donc de coût 0.

Il est évident qu'un sous-graphe pourrait se retrouver plusieurs fois dans le graphe cible, ce qui est souvent le cas si leur nombre de sommets est limité (le graphe n'est pas assez discriminant). Les étiquettes assignées aux sommets et/ou arêtes peuvent alors servir à différencier un bon appariement d'un mauvais.

Quant à la complexité de l'isomorphisme de sous-graphe, il s'agit d'un problème difficile [Garey et Johnson, 1979] :

Complexité de l'isomorphisme de sous-graphe Le problème d'isomorphisme de sous-graphe est NP-complet.

La plupart des méthodes et algorithmes, utilisant des heuristiques, cités dans la partie précédente (3.2.1) s'appliquent également à l'isomorphisme (tolérant) de sous-graphe,

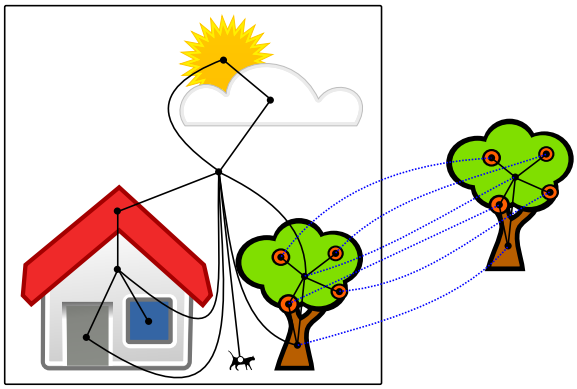


FIG. 3.x – Isomorphisme de sous-graphe entre un motif (à droite) recherché dans une image (à gauche). L'appariement est représenté par les arcs bleus.

comme les arbres de recherche [Ullmann, 1976]. On peut également citer l'approche par graphe d'association [Levi, 1972], où chaque appariement de deux sommets est dénoté par un sommet dans un nouveau graphe, dont les arêtes dénotent les compatibilités d'appariements.

Concernant les graphes planaires, comme pour le cas de l'isomorphisme, il a été montré que l'isomorphisme de sous-graphe pouvait se résoudre en temps linéaire, mais là aussi avec une constante rendant son utilisation pratique difficile [Eppstein, 1995].

3.2.3 Plus grand sous-graphe commun

Une autre possibilité est de rechercher ce qu'ont en commun deux images, et plus précisément quelle est leur plus grande partie commune. Il s'agit, dans la théorie des graphes, de la recherche de plus grand sous-graphe commun.

Définition 3.11 (Sous-graphe commun)
Soient G, G_1 et G_2 trois graphes. G est un sous-graphe commun *à G_1 et G_2 s'il existe un isomorphisme de sous-graphe entre G et G_1 d'une part, et G et G_2 d'autre part.*

Définition 3.12 (Plus grand sous-graphe commun)
*Un sous-graphe commun $G = (V, E)$ à G_1 et G_2 est maximal (*i.e. *il s'agit du* plus grand sous-graphe commun*) s'il n'existe pas d'autre sous-graphe commun $G' = (V', E')$ à G_1 et G_2, tel que $|V'| > |V|$.*

Un plus grand sous-graphe commun de G et G' est un graphe G'' qui est un sous-graphe de G et G' et qui, parmi tous les sous-graphes possible de G et G', a le plus grand nombre de sommets. Notons que le plus grand sous-graphe commun de deux graphes peut ne pas être unique. Le problème du plus grand sous-graphe commun peut lui aussi s'interpréter en terme d'appariement : il correspond à un appariement univoque de taille maximale en le nombre de sommets, avec une contrainte dure sur les sommets et les

arêtes. La figure 3.xi illustre l'appariement d'un plus grand sous-graphe commun de deux graphes.

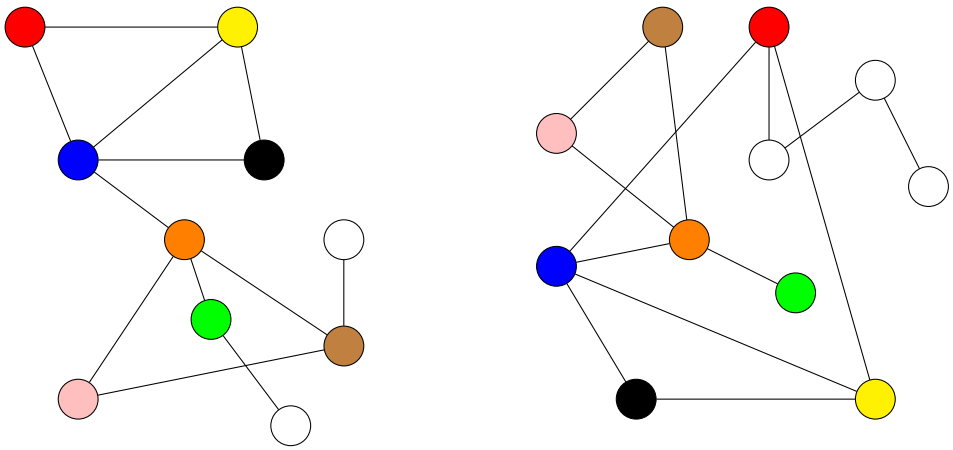


FIG. 3.xi – Plus grand sous-graphe commun : l'appariement est dénoté par les couleurs.

Complexité du plus grand sous-graphe commun Le problème de recherche du plus grand sous-graphe commun est NP-difficile.

Le plus grand sous-graphe commun a été utilisé pour définir une distance entre graphes [Bunke et Shearer, 1998]. Soient $G = (V, E)$ et $G' = (V', E')$ deux graphes, de plus grand sous-graphe commun $gsc(G, G')$, alors leur distance $d(G, G')$ vaut

$$d(G, G') = 1 - \frac{|gsc(G, G')|}{max(|V|, |V'|)}. \tag{3.5}$$

Cette formule a été la base d'une étude calculant la distance entre deux graphes en passant par leur union [Wallis et al., 2001].

3.3 Distance d'édition de graphes et similarité entre images

Allons plus loin. Il est peu fréquent, dans des cas où les graphes sont plus complexes, ce qui correspond souvent à la réalité, qu'ils soient parfaitement isomorphes. Même si les images sont très proches, certaines petites différences peuvent modifier la structure des graphes, comme l'illustre la figure 3.xii. En outre, les images du monde réel sont souvent bruitées (changement de conditions de prise de vue...), et les graphes extraits en sont donc modifiés. Pour cette raison, il est généralement nécessaire d'introduire une certaine tolérance [Bunke, 1998] lors de l'appariement des graphes (par exemple, on peut imaginer des isomorphismes à k sommets près, à m arêtes près...). Un grand nombre de méthodes des références déjà citées dans ce chapitre résolvent en fait un

isomorphisme approximatif : à défaut de trouver un isomorphisme parfait (parce qu'il n'existe pas, ou parce qu'il est trop fastidieux à trouver), les solutions renvoyées sont des appariements tolérants optimaux (qui trouvent la meilleure solution, dans le pire des cas en temps exponentiel), ou suboptimaux (qui trouvent un minimum local, en général en temps polynomial). Ceci peut être rapproché du calcul de distances entre graphes.

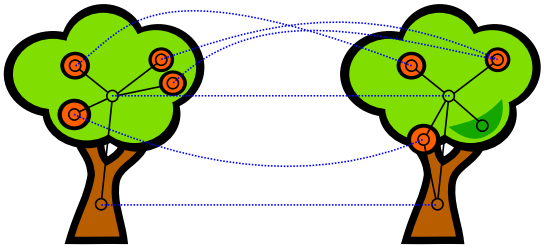


FIG. 3.xii – Ces deux graphes ne sont plus isomorphes. Pourtant : ils sont proches, il existe entre eux un isomorphisme tolérant.

Nous avons évoqué, dans le chapitre 1 l'existence d'une mesure de distance entre chaînes, la distance d'édition. Nous avons ajouté qu'elle avait été adaptée pour permettre de calculer la distance entre arbres. Ce fut également le cas pour la distance entre graphes. Ainsi, on définit un ensemble d'opérations permettant de passer d'un graphe à l'autre [Sanfeliu et Fu, 1983]. En général, il s'agit de la suppression, de l'insertion ou de la substitution d'un sommet ou d'une arête. À chacune de ces opérations est assignée un coût, qui peut être toujours le même, ou dépendre des sommets et arêtes considérés. La suite des opérations d'édition permettant de passer d'un graphe à l'autre est appelé script d'édition. Il est évident qu'il existe une infinité de scripts, mais la distance d'édition correspond à celui de coût minimal :

Définition 3.13 (Distance d'édition de graphes)
La distance d'édition de deux graphes G_1 *et* G_2 *correspond au coût du script d'édition optimal.*

La valeur de la distance nous informe sur la dissemblance entre les deux graphes : plus elle est petite et plus ils sont similaires. La distance d'édition de graphe est en fait un appariement univoque avec des contraintes souples sur les sommets et les arêtes. Dans le domaine de l'image (voir la figure 3.xiii), cela revient à se demander comment modifier le graphe de gauche pour obtenir celui de droite. Parmi toutes les solutions possibles, une se trouve être la moins coûteuse (plusieurs solutions optimales peuvent exister).

Bunke [Bunke, 1997] a montré que, dans le cas où toutes les opérations d'édition ont un coût unitaire (la distance est alors égale à la longueur du plus petit script d'édition possible), distance d'édition et plus grand sous-graphe commun sont intimement liés. En effet, si $G'' = (V'', E'')$ est le plus grand sous-graphe commun de $G = (V, E)$ et

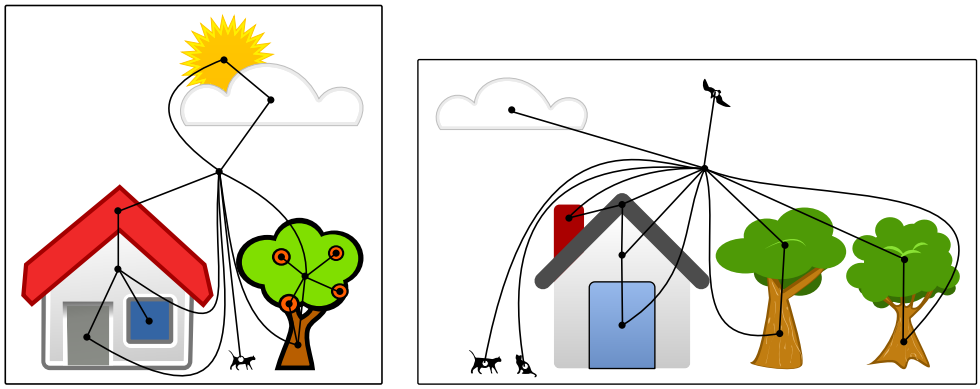


FIG. 3.xiii – Distance d'édition de graphes : quelles opérations pour passer de l'un à l'autre ?

$G' = (V', E')$, alors leur distance d'édition $d_e(G, G')$ est la suivante

$$d_e(G, G') = |V| + |V'| - 2|V''|. \tag{3.6}$$

On peut également rapprocher isomorphisme de sous-graphe et distance d'édition [Zeng et al., 2009]. En effet, étant donné deux graphes $G = (V, E)$ et $G' = (V', E')$, avec $|V| < |V'|$ et $|E| < |E'|$, alors G est un sous-graphe de G' si et seulement si

$$d_e(G, G') = (|E'| - |E|) + (|V'| - |V|). \tag{3.7}$$

Dans ce cas, les opérations d'édition ont aussi un coût unitaire. Ainsi, on peut utiliser la distance d'édition pour résoudre l'isomorphisme de sous-graphe, que nous avons déjà dit être NP-complet. Par conséquent, la complexité de la distance d'édition de graphes est la suivante :

Complexité de la distance d'édition de graphes Le calcul de la distance d'édition de deux graphes est NP-difficile.

Les travaux portant sur la distance d'édition de graphes font donc appel à des heuristiques. Dans [Riesen et al., 2007], les auteurs cherchent tout d'abord un appariement optimal des sommets, avant d'y ajouter le coût des arêtes (la solution finale est donc suboptimale), en utilisant l'algorithme d'assignement des sommets de Munkres [Munkres, 1957] basé sur une matrice de coûts. Ils est aussi possible de faire appel à des arbres de recherche [Neuhaus et al., 2006], où les auteurs adaptent l'algorithme A^* [Hart et al., 1968] (construction dynamique d'un arbre de recherche avec à chaque étape sélection du sommet le moins coûteux) ou a des probabilités [Myers et al., 2000]. Certains auteurs ont également proposé de nouvelles opérations d'édition, comme la fusion de deux sommets ou la division d'un sommet en deux [Ambauen et al., 2003, Berretti et al., 2004] (ceci trouve tout son sens lorsque les graphes représentent des images, pour pallier des problèmes de sur ou sous-segmentation). D'autres encore ont étudié l'apprentissage des coûts d'édition [Neuhaus et Bunke, 2007].

4

Une méthode d'extraction de graphes plans à partir d'images

Sommaire

Dans le premier chapitre de cette thèse (se référer aux parties 1.2 et 1.3), nous avons pu découvrir pourquoi il était utile de chercher à représenter les images numériques sous une autre forme qu'une simple matrice de pixels, et nous avons exposé certaines de ces représentations alternatives : les histogrammes, les chaînes, les arbres et les graphes. Les graphes, que nous privilégions dans cette thèse, sont en effet riches en information, mais constituent également des modèles symboliques liés à des problématiques telles l'isomorphisme de graphes, qui trouvent tout leur sens dans un domaine d'application tel que l'image (voir le chapitre 3). De même, nous avons pu constater, dans le chapitre 2, que l'appariement de graphes, et les fonctions de coût associées, permettaient eux aussi de travailler dans ce sens.

Dans ce chapitre, nous présenterons notre méthode d'extraction de graphes à partir d'images numériques. Après avoir exposé le cahier des charges associé, nous détaillerons les différentes étapes permettant d'obtenir ces graphes. Nous montrerons également que ces graphes permettent de coder une certaine version des images, avec moins d'information qu'à l'origine. Nous terminerons par des expérimentations visant à analyser les pertes engendrées par l'extraction de graphes, à nous comparer à d'autres méthodes, et

nous porterons également notre attention sur la taille de ces graphes. Nous conclurons ce chapitre sur une discussion.

4.1 Finalité et moyens

Deux étapes de prétraitement d'une image peuvent mener à la construction d'un graphe : la segmentation et l'extraction de caractéristiques (ou points d'intérêt). Pour la première, on obtient par exemple un RAG (voir la partie 1.3.1). Outre les inconvénients de ce genre de modèles cités dans la partie correspondante, on peut ajouter que la structure du graphe ainsi obtenu n'a le plus souvent que peu de rapport avec ce que représentait l'image à l'origine, surtout si la segmentation est grossière. D'un autre côté, si l'on construit un graphe par extraction de points d'intérêt qui seront ensuite reliés entre eux, il est possible que certains points engendrent du bruit dans le graphe, ou encore que l'on doive en extraire un trop grand nombre pour être certain d'avoir tous ceux qui sont important. Ceci engendre des graphes de trop grande taille, chose néfaste pour des problèmes complexes comme l'isomorphisme. De plus, associer un sommet par pixel (voire même groupe de pixels) pose un problème de géométrie discrète, puisqu'ils ont dans ce cas une épaisseur.

Demandons-nous alors ce que doit être un *bon* graphe. Premièrement, il doit pouvoir être extrait de tout type d'images, et être robuste : si l'image est un peu déformée (par exemple par modification légère de la prise de vue, ou de l'échelle, une occultation raisonnable, un changement de luminosité ou bien une modification d'autres paramètres), le graphe doit, lui, rester le plus stable possible. En outre, la sémantique de l'image devrait être respectée par le graphe. Ceci signifie qu'une fois dessiné, le graphe soit bien une représentation des objets de l'image (par exemple, que les frontières des objets de l'image coïncident avec les arêtes du graphe) : une personne n'ayant pas vu l'image doit reconnaître, sur le dessin du graphe, ses principaux objets ou au moins les deviner. En ce sens, les faces du graphe auront toute leur importance. De plus, il doit respecter certaines propriétés pour être raisonnable vis-à-vis de certains algorithmes : les problèmes d'isomorphisme de graphes, et ceux, liés, d'isomorphisme de sous-graphes, de plus grand sous-graphe commun et de distance d'édition de graphes (voir le chapitre 3) devraient être solvables en temps polynomial. Enfin, nous souhaiterions être en possession de graphes à partir desquels il serait possible de reconstruire l'image associée, et que, de plus, cette image soit une bonne approximation de l'image d'origine. Cette perte est en outre nécessaire, pour permettre la généralisation : des images similaires doivent avoir des graphes très proches. De plus, cette vision peut être rapprochée du principe du *minimum description length* [Rissanen, 2007], régit par le fait que la meilleure hypothèse représentant un ensemble de données est celle qui mène à la meilleure compression.

Nous répondrons à ce cahier des charges de la façon suivante :

- pour obtenir une certaine robustesse, nous ferons appel non à une seule opération de traitement d'images, mais à deux : nous combinerons à cet effet la segmentation et l'extraction de caractéristiques ;
- combiner ces deux prétraitements assurerait l'obtention de graphes sémantique-

ment semblables aux images d'origine [Pardo, 2002, Jiten et Mérialdo, 2007] ;
- pour que le graphe soit algorithmiquement raisonnable, nous limiterons sa taille (donc son nombre de sommets) tout en respectant les deux points pré-cités, et nous ajouterons deux contraintes qui le rendront fort intéressant : il devra être plan, et connexe ;
- enfin, pour pouvoir reconstruire une image, nous associerons aux graphes quelques données supplémentaires, outre les sommets et les arêtes.

Dans la suite de ce chapitre, nous utiliserons les notations suivantes :
- I désignera une image numérique ;
- $p(I, i, j)$ désignera un pixel de l'image I, de coordonnées i et j ;
- I_r sera l'image de la segmentation de I en r régions (il s'agit d'une image dont chaque région a une couleur correspondant à la moyenne des couleurs de ses pixels, comme nous le verrons dans la partie 4.3) ;
- $G(I_r, p)$ sera le graphe obtenu en extrayant p pixels d'intérêt sur les contours de segmentation de l'image I_r ;
- P sera la fonction de *position* qui, à tout sommet d'un graphe associe ses coordonnées spatiales ;
- C sera la fonction de *couleur* qui, à toute face d'un graphe associe sa couleur ;
- $I(G(I_r, p), P, C)$ désignera l'image de la reconstruction du graphe $G(I_r, p)$ à l'aide des données de P et C (nous traiterons le cas particulier des pixels traversés par les arêtes du graphe dans la partie 4.3).

De plus, nous analyserons la perte engendrée par les différentes étapes de notre méthode d'extraction de graphes plans à partir d'images. Il s'agira de comptabiliser les différences entre les images : image originelle I, image de la segmentation en r régions I_r et image du graphe de p pixels d'intérêt $I(G(I_r, p), P, C)$. Pour cela, nous définissons une distance d_p entre tout pixel p_1 d'une image I_1 et tout pixel p_2 d'une image I_2 comme étant la distance normalisée entre leurs vecteurs de couleur. Ainsi, si la couleur de p_1 dans I_1 est $c_{I_1}(p_1) = (r_1, v_1, b_1)$ et la couleur de p_2 dans I_2 est $c_{I_2}(p_2) = (r_2, v_2, b_2)$ (ici nous prenons donc le codage RVB), alors :

$$d_p(p_1, p_2) = \frac{|r_2 - r_1| + |v_2 - v_1| + |b_2 - b_1|}{3 * 255}$$

Et nous définissons, pour toutes images I_1 et I_2 de définition $n \times m$ pixels :

$$perte(I_1, I_2) = \frac{\sum_{i=0}^{i=n-1} \sum_{j=0}^{j=m-1} d_p(p(I_1, i, j), p(I_2, i, j))}{n \times m}$$

Revenons pour terminer sur le problème évoqué en début de partie, portant sur la correspondance entre pixels et sommets du graphe. En effet, si un pixel est désigné pour devenir sommet du graphe, cela pose un problème de géométrie discrète : un sommet n'a pas d'épaisseur, un pixel si. Pour pallier cela, nous utiliserons par la suite la terminologie et certains objets de l'interpixel [Françon, 1996, Fiorio, 1995]. En effet, considérer une image comme une matrice de pixels pose certains problèmes. Notamment, si l'on parle de frontières entre deux régions, désigne-t-on des pixels ? Si oui, lesquels ? Intuitivement,

par frontière, on désignerait plutôt des lignes entres les régions, donc entre les pixels. On dira ainsi que les pixels sont séparés par des lignels, et que les extrémités des lignels sont des pointels (pour une illustration, on pourra se référer à la figure 4.i). Il existe donc quatre lignels par pixel, et deux pixels adjacents au sens du 4-voisinage ont un lignel (et donc deux pointels) en commun.

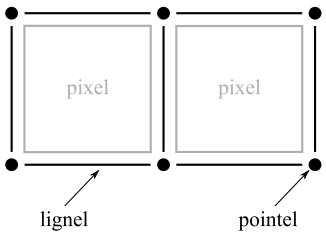


FIG. 4.i – Terminologie de l'interpixel.

4.2 Construction du graphe

Dans cette partie, nous détaillons les différentes étapes de notre méthode de construction d'un graphe à partir d'une image. Celle-ci se base sur les deux opérations usuelles de traitement d'images introduites dans la partie 1.1.2 : la segmentation et l'extraction de pixels d'intérêt. Combiner ces deux opérations permet de tirer profit de leurs avantages respectifs et d'obtenir ainsi des graphes préservant l'information sémantique et topologique contenue dans l'image originale (c'est-à-dire une représentation cohérente vis-à-vis de la disposition spatiale des objets représentés sur l'image, leurs frontières, et leurs relations d'adjacence ou d'inclusion). De plus, ces graphes constituent alors une approximation valide de l'image originelle qui minimise la fonction de perte présentée dans la partie 4.1. De façon plus formelle, la méthode d'extraction permet d'obtenir pour toute image I, un graphe G et des fonctions P et C, tels que $perte(I, I(G, P, C))$ soit minimisée.

Dans la suite, nous illustrerons nos propos en considérant le manchot de la figure 4.ii, qui servira d'exemple, tout au long de cette section, à la construction de graphes. Elle provient de la base de données d'images segmentées de Berkeley[1], que nous utiliserons également pour nos expérimentations (partie 4.4).

4.2.1 Segmentation de l'image

Le processus de segmentation d'une image, détaillé dans la partie 1.1.2, vise à définir un ensemble de régions disjointes, constituées de pixels homogènes selon un critère prédéfini. De nombreuses segmentations sont possibles pour une même image, comme on peut le voir sur la figure 4.iii.

[1] http://www.eecs.berkeley.edu/Research/Projects/CS/vision/grouping/segbench/

FIG. 4.ii – Image originelle dont on cherche à extraire un – ou des – graphe(s).

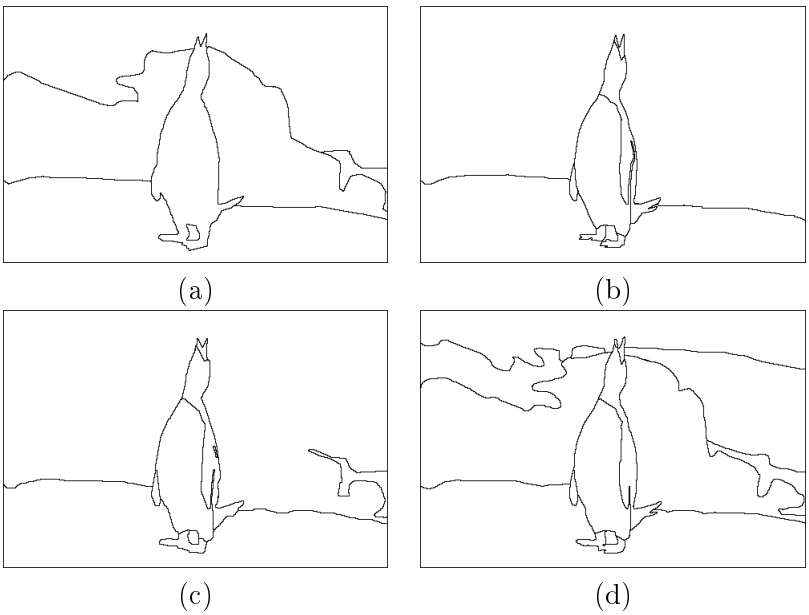


FIG. 4.iii – Différentes segmentations possibles d'une même image. Les bordures des régions sont délimitées par des pixels noirs. (a) 7 régions, (b) 10 régions, (c) 12 régions et (d) 18 régions.

L'information essentielle fournie par ce processus est directement liée à ces régions : chacune d'entre elles a une dimension sémantique, en ce qu'elle représente un des concepts, ou une partie d'un des concepts, figurant sur l'image originale. C'est cette dimension sémantique que l'on cherche tout d'abord à préserver lors de l'extraction du graphe. Intuitivement, la signification des régions est conservée si les arêtes du graphe concordent au mieux avec les bordures des régions. Celles-ci sont composées des pixels adjacents au complément de chacune des régions.

Pour atteindre ce but, nous nous intéressons aux pointels et lignels introduits dans la partie 4.1 : les quatre coins d'un pixel sont appelés pointels, et chaque couple de pixels adjacents est séparé par un lignel (pour un rappel, voir la figure 4.i). Si ces deux pixels n'appartiennent pas à la même région, on peut considérer que le lignel séparateur constitue une frontière. Une région est alors délimitée par une suite de lignels de frontière, qui doivent être consécutifs.

Ainsi, considérons la figure 4.iv, qui est issue de la segmentation en sept régions distinctes numérotées de l'image originelle du manchot. Les lignels séparant des pixels de régions différentes y sont représentés, tout comme ceux correspondant aux bords de l'image (pour les représenter graphiquement, il a fallu leur assigner une épaisseur, alors que, pour rappel, ils n'en ont pas en réalité). Chacune des régions peut alors être décrite en terme de lignels consécutifs, qui forment des cycles. Ainsi, les régions 1, 2, 3, 4, 5 et 7 sont chacune décrites par un seul cycle de lignels. Cependant, il arrive également que les régions ne soient pas connexes, et que certaines soient incluses les unes dans les autres : c'est le cas de la 7, incluse dans la 6. Dans ce cas, la région dite englobante (*i.e.*, dans le cas présent, la 6), est décrite par plusieurs cycles de lignels (ici, deux cycles), un pour la région englobante, les autres pour la ou les régions englobées (ici, uniquement la 7).

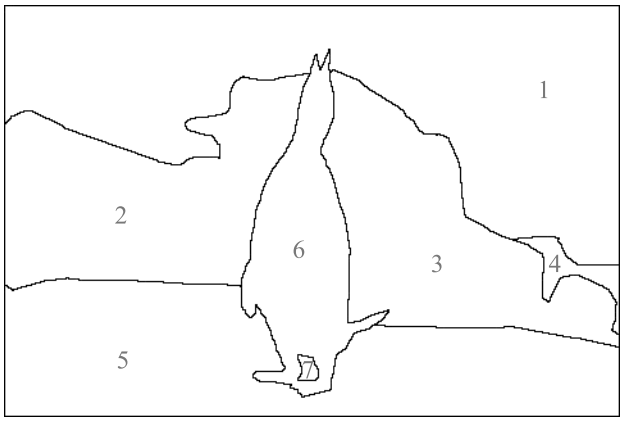


Fig. 4.iv – Ensemble de lignels dépeignant les frontières des régions obtenues par segmentation.

Il faut désormais trouver les sommets du graphe nous permettant de respecter au mieux les lignels. Spontanément, la solution serait de considérer que chaque pointel

appartenant à un lignel séparant deux régions différentes constitue un sommet du graphe (et tout lignel deviendrait alors arête). Cependant, le graphe ainsi obtenu contiendrait un nombre assez conséquent de sommets, ce qui n'est pas recommandé pour des raisons de combinatoire (notamment pour les problèmes d'isomorphismes de graphes évoqués dans le chapitre 3). Inversement, un nombre trop faible de sommets engendrerait des arêtes formant des segments trop éloignés des frontières des régions : l'approximation de l'image originelle serait trop importante, ce qui est rédhibitoire pour des tâches de reconnaissance de formes, par exemple. Par conséquent, il s'agit de choisir les sommets de façon à compresser avec le moins de perte possible l'information fournie par les lignels (qui se trouve être déjà une compression de l'image originelle) : en effet, ce n'est pas la configuration exacte des régions, pixel par pixel, qui nous est utile, mais leur forme globale.

La question se pose désormais en ces termes : comment bien approximer les régions, pour que la perte soit minimale ? De nombreuses approximations d'une région sont possibles, et ceci correspond notamment à des travaux de simplification polygonale [Bhowmick et Bhattacharya, 2007, Damiand et Coeurjolly, 2008]. Nous souhaitons privilégier une méthode garantissant une faible distance par rapport à la véritable frontière de la région, tout en produisant un graphe de taille raisonnable. Notre postulat est que ce but sera atteint si l'on choisit comme sommets du graphe un sous-ensemble des pointels des lignels séparateurs de régions, et ce avec l'aide d'un extracteur de points d'intérêt.

4.2.2 Extraction de pixels d'intérêt et affinage

Pour sélectionner les pointels qui constitueront les sommets du graphe, nous faisons appel à la seconde opération de traitement d'images présentée dans la partie 1.1.2 : la détection de pixels d'intérêt. Bien qu'usuellement directement appliqués sur les images originelles, nous choisissons d'exécuter les détecteurs de pixels d'intérêt sur les images binaires représentant les bordures des régions (comme on a pu le voir sur la figure 4.iii). Un tel choix se justifie pour plusieurs raisons. Tout d'abord, ceci permet de supprimer totalement la détection de pixels d'intérêt dus à du bruit sur l'image. Ce choix nous assure également de ne pas extraire de pixels internes aux régions et correspondant par exemple à des zones de fort contraste, des coins ou encore à des zones fortement texturées. Enfin, cela nous garantit d'obtenir des pixels assez bien répartis sur l'ensemble des bordures des régions et ainsi de représenter chacune d'entre elles. On obtient donc un ensemble de pixels d'intérêt situés directement sur, ou très proches, de la frontière des régions. La figure 4.v permet de comparer l'extraction de pixels d'intérêt sur l'image originelle, l'image segmentée en 7 régions, et en 18 régions.

Cependant, malgré ces précautions, cet ensemble de pixels d'intérêt n'est pas encore satisfaisant, et cela pour deux raisons. Premièrement, il est possible que certains des pixels extraits ne soient pas réellement intéressants pour l'usage que l'on souhaite en faire, s'ils sont trop éloignés des frontières des régions. Il faut donc supprimer ceux qui vérifient ce critère. Pour cela, nous prenons en considération, pour chaque pixel détecté, un masque formé de lui-même et de ses huit voisins (ou de ses cinq ou trois voisins si

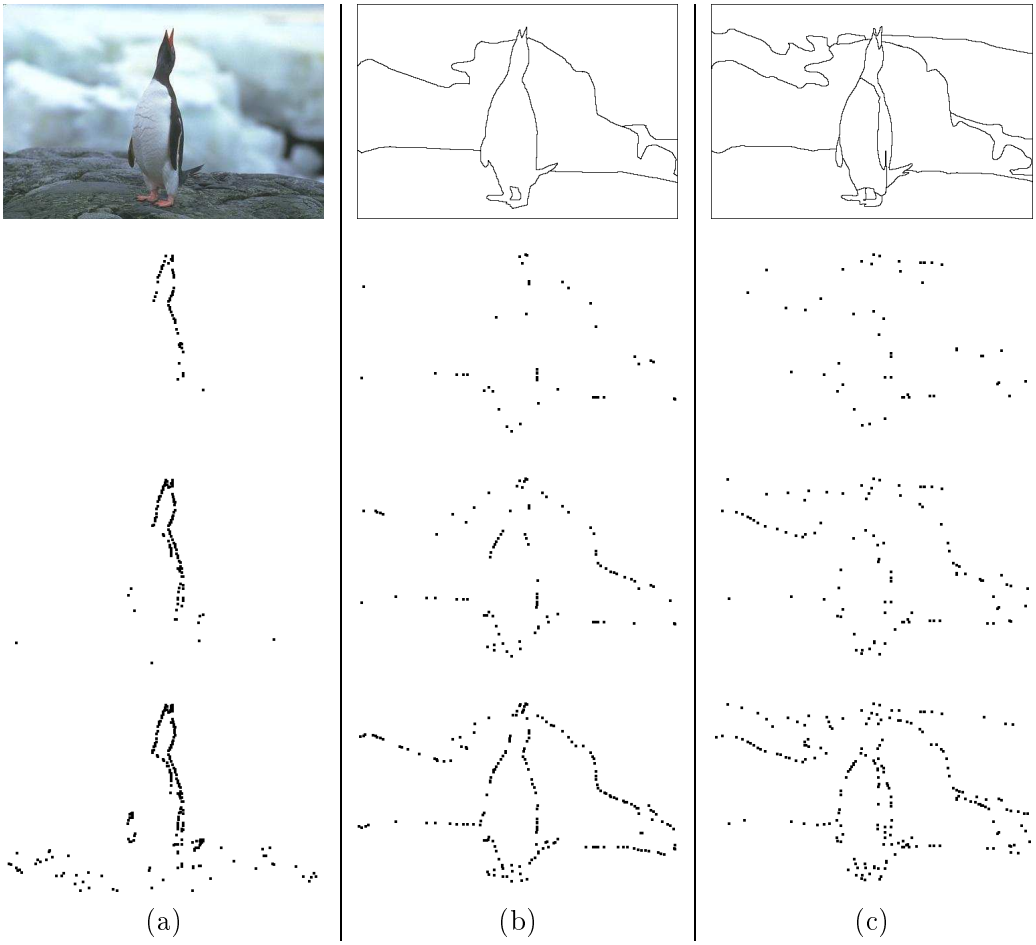


FIG. 4.v – Extraction de 50, 100 et 200 pixels d'intérêt. Colonne (a), sur l'image originelle, seule la zone du manchot est bien représentée. Colonnes (b) et (c) sur des images segmentées : dès 100 pixels, toutes les régions sont discernables.

le pixel est sur un bord ou un coin de l'image). S'il s'avère que ce masque ne contient que des pixels appartenant à la même région (*i.e.* le masque est uniforme), alors le pixel détecté correspondant est supprimé de l'ensemble des pixels d'intérêt. La seconde étape d'affinage de l'ensemble des pixels d'intérêt vise à assurer que toutes les intersections entre régions soient correctement représentées. On va donc potentiellement ajouter des pixels à l'ensemble. L'avantage de cet ajout est bien sûr de suivre les frontières des régions, mais il existe également un bénéfice majeur qui sera explicité dans la section suivante 4.2.3. Cette fois-ci, on analyse le masque de chacun des pixels de l'image, et on ajoute comme pixel d'intérêt tout pixel dont le masque recouvre au moins trois régions (ou au moins deux, si le pixel appartient au bord de l'image). Pour terminer, nous ajoutons également les quatre coins de l'image. Un exemple de résultat d'extraction de pixels d'intérêt et de leur affinage est représenté sur la figure 4.vi.

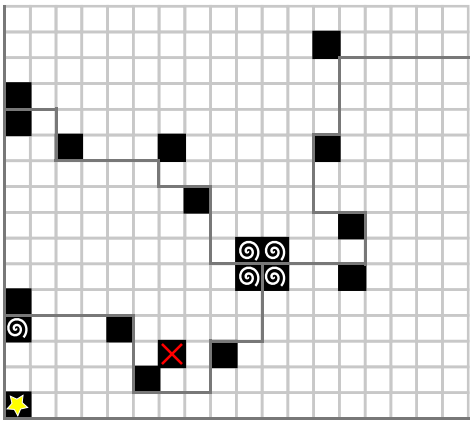


FIG. 4.vi – Extraction et affinage des pixels d'intérêt le long des bordures d'une partie d'une image segmentée. Aux pixels détectés automatiquement (en noir), on ajoute les pixels d'intersection manquants (spirales blanches) et les coins de l'image (étoile). On supprime également les masques uniformes (croix rouge).

4.2.3 Des pixels aux pointels

Nous sommes désormais en possession d'un ensemble de pixels d'intérêt adéquat. La prochaine étape consiste en la sélection, parmi les quatre pointels possibles pour chaque pixel d'intérêt, des plus pertinents. Nous les appellerons par la suite pointels d'intérêt. La règle de sélection est la suivante :

1. on considère chaque pointel π de chaque pixel d'intérêt indépendamment ;
2. soient p_0, p_1, p_2 et p_3 les quatre pixels ayant en commun le pointel π (numérotés en tournant autour du pointel) ;
3. si $\exists p_i$ tel que $c_I(p_{i-1}) \neq c_I(p_i)$ et $c_I(p_i) \neq c_I(p_{i+1})$ (modulo 4) alors π est un pointel d'intérêt. Rappelons que $c_I(p)$ est la couleur du pixel p de l'image I ; ici l'image est segmentée, la couleur est donc en fait la région d'appartenance du pixel.

En fait, il s'agit de conserver en tant que pointels d'intérêt les pointels dont les régions des pixels adjacents ont des configurations en diagonale, ou en coins. Des illustrations de tels cas sont visibles sur la figure 4.vii.

Les pointels d'intérêt forment dès lors les sommets du graphe. Le résultat d'une telle sélection est visible sur la figure 4.viii : les pointels blancs sont les pointels d'intérêt.

Il s'agit maintenant de créer les arêtes du graphe, et cela en reliant les sommets tout en suivant les bordures des régions pour respecter leur forme. La figure 4.ix représente, en noir, les arêtes du graphe et, en blanc, ses sommets. L'approximation des bordures des régions, en gris foncé, est alors visible. De plus, chaque face du graphe correspond bien à une région de la segmentation (si des régions sont décrites par plusieurs cycles de lignels, elles sont alors décrites par plusieurs faces : une englobante, et une ou plusieurs englobées).

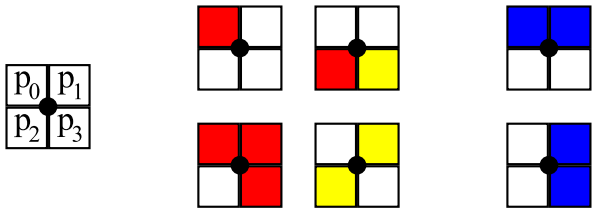


FIG. 4.vii – Règle de sélection des pointels d'intérêt : à gauche, les 4 pixels adjacents au pointel considéré. Au centre, des exemples de configuration dénotant un pointel d'intérêt. À droite, des exemples de configuration ne dénotant pas un pointel d'intérêt.

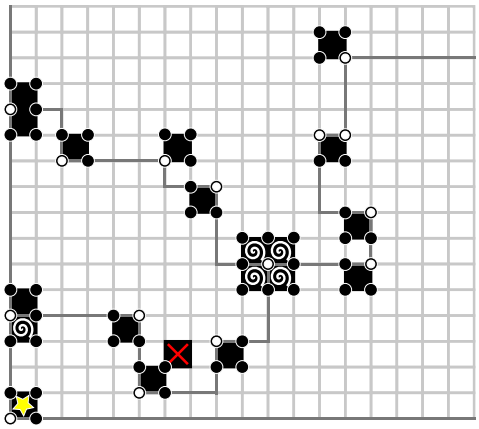


FIG. 4.viii – Des pixels d'intérêt aux pointels d'intérêt : tous les pointels possibles (ronds noirs et ronds blancs), et les 15 pointels d'intérêt conservés (ronds blancs).

Revenons maintenant sur l'avantage primordial apporté par l'ajout des pixels d'intersection évoqué dans la partie 4.2.2. Reconsidérons la figure 4.viii et imaginons que nous n'ayons pas ajouté les pixels d'intersection non détectés automatiquement (pixels avec une spirale blanche). Certains pointels d'intérêt peuvent alors ne plus être sommets du graphe (dans l'exemple, il s'agit du seul pointel central aux quatre pixels d'intérêt à spirale du centre de l'image). La première conséquence est évidemment que l'approximation des régions est dès lors plus importante. Mais la répercussion la plus remarquable est représentée sur la figure 4.x : il peut y avoir création de nouvelles faces dans le graphe (des triangles ou des quadrilatères) au niveau de l'intersection des régions. Sur l'image, c'est le cas du triangle grisé. Ces nouvelles faces ont le double désavantage de ne pas appartenir à une région propre et, ainsi, de n'avoir aucun sens sémantique. Elles ne sont donc pas souhaitables en regard du cahier des charges que nous nous sommes fixé dans la partie 4.1 pour extraire des graphes à partir d'images. L'ajout de tous les pixels d'intersection nécessaires à une bonne représentation des régions est donc essentiel.

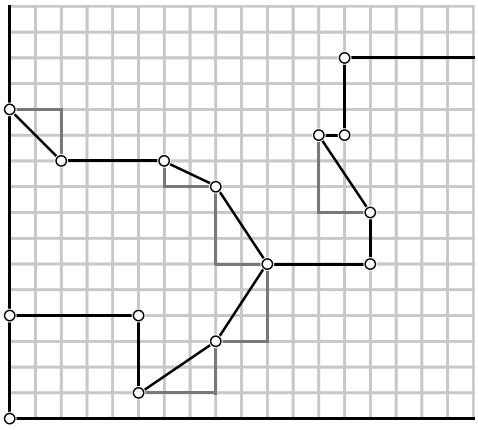


FIG. 4.ix – Les pointels d'intérêt sont reliés en suivant les bordures des régions, formant ainsi les arêtes du graphe. L'approximation des bordures est alors visible.

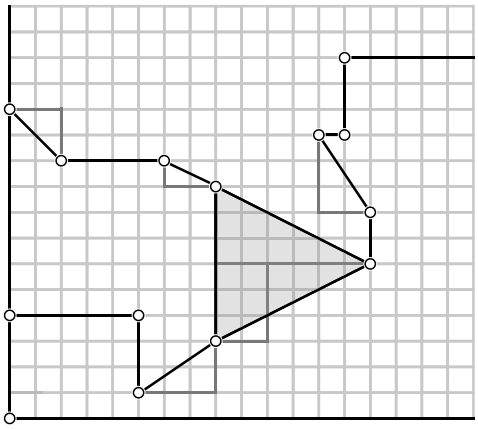


FIG. 4.x – Si les pixels d'intersection n'étaient pas tous détectés : création de faces dans le graphe n'appartenant pas à une région donnée, et n'ayant aucun sens sémantique (ici, en gris).

La dernière remarque de cette partie concerne la taille des graphes obtenus. Comme nous l'étudierons en détail par des expérimentations dans la partie 4.4.3, elle est intimement liée au nombre de pixels d'intérêt. On peut ainsi extraire des graphes de taille plus ou moins grande, et qui approximent plus ou moins les régions de la segmentation (en toute logique, moins le nombre de pixels d'intérêt détectés est important et plus les régions sont approximées). Des exemples de graphes de différentes tailles obtenus d'après la segmentation de la figure 4.iv sont représentés sur la figure 4.xi. On remarque que la région 7, qui est incluse dans la 6 (corps du manchot), n'est décelée, avec le détecteur utilisé, que lorsque la taille du graphe devient plus importante (dans le cas présent, pour 200 pixels d'intérêt extraits, soit un graphe de taille 334).

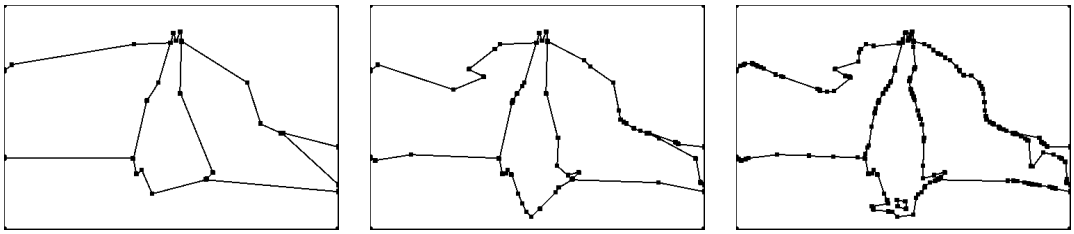


FIG. 4.xi – Plusieurs graphes obtenus d'après la même segmentation d'une image, en faisant varier le nombre de pixels d'intérêt extraits. De gauche à droite, on a respectivement 35, 80 et 200 pixels extraits, auxquels ont été ajoutés par affinage 43, 43 et 41 pixels d'intersection et/ou coins de l'image, pour arriver à 81, 145 et 334 pointels d'intérêt formant les sommets des graphes.

4.2.4 Triangulation par région

Nous sommes dorénavant en possession de graphes qui peuvent être considérés comme des ensembles de faces représentant des objets ou parties d'objets. Ces graphes ont notamment pour caractéristique, et non des moindres, d'être des représentations planes de graphes planaires. Nous souhaitons néanmoins leur apporter plus de robustesse, tout en conservant cette propriété importante de planarité.

Par augmenter la robustesse, nous entendons renforcer la structure des graphes. Par là-même, puisque nos graphes doivent pouvoir répondre à des problématiques de reconnaissance de formes, il s'agit de renforcer la structure des faces, qui sont les objets. Dans ce but, nous choisissons d'insérer des arêtes supplémentaires dans chacune des faces, en faisant appel à la triangulation de Delaunay (plus de détails sur cette structure géométrique sont disponibles dans l'annexe A). Cette triangulation est unique pour un ensemble de points donné. Ainsi, même si la position des sommets est légèrement modifiée à cause de transformations appliquées à l'image originelle, la triangulation, elle, restera stable ; c'est-à-dire que les deux graphes correspondant à ces triangulations seront les mêmes : ils auront les mêmes sommets et les mêmes arêtes. De plus, confrontée aux problèmes évoqués d'isomorphismes de graphes, la triangulation permettra d'éviter de mettre en relation des faces ayant des bordures similaires, mais des formes différentes.

Ceci est illustré par la figure 4.xii : les deux graphes de la première ligne sont isomorphes. Pourtant, leur forme est très différente, et cela n'aurait pas de sens de les apparier dans une application de reconnaissance de formes. La ligne en-dessous représente ces mêmes graphes, auxquels a été appliquée une triangulation de Delaunay : ils ne sont alors plus isomorphes.

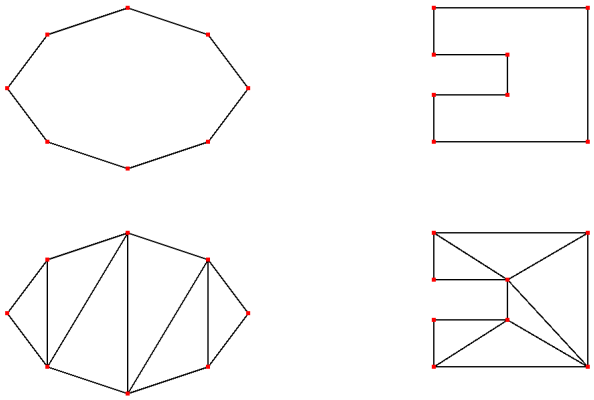


FIG. 4.xii – Sur la première ligne, deux graphes isomorphes, qui ne le sont plus lorsqu'une triangulation de Delaunay leur est appliquée (seconde ligne).

Sur cette même figure, on remarque que le graphe de droite, de par sa représentation plane, est concave. Pour obtenir la triangulation de Delaunay représentée, nous avons choisi de supprimer les arêtes n'étant pas à l'intérieur de la face. En effet, la triangulation de Delaunay engendrant des graphes convexes, certaines arêtes peuvent alors ne plus appartenir à l'intérieur des faces. La sémantique de ces faces en serait ainsi modifiée, voire perdue. Il est donc nécessaire d'ajouter à notre méthode une phase de correction de la triangulation pour éliminer de telles arêtes. Le processus de triangulation des graphes se déroule donc en deux étapes : la première met en jeu une triangulation face par face, dont les arêtes extérieures aux faces sont supprimées lors de la seconde étape. En fusionnant l'ensemble des faces triangulées, on obtient le graphe final, comme on peut le voir sur la figure 4.xiii.

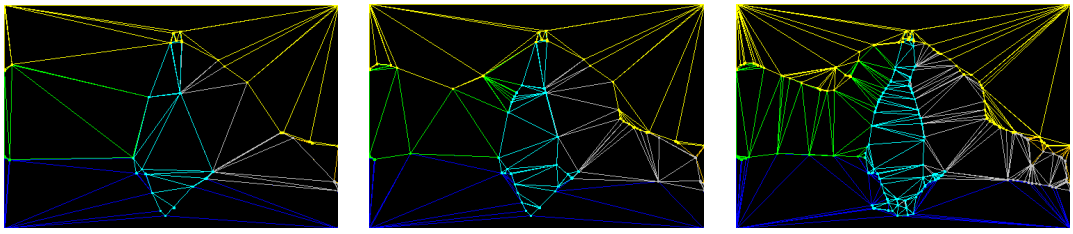


FIG. 4.xiii – Triangulation des graphes de la figure 4.xi : 145 et 334 sommets.

L'insertion des arêtes de la triangulation apporte un second avantage. On a remarqué que certaines régions de la segmentation pouvaient être incluses les unes dans les autres. Il en découle que le graphe que l'on obtient en suivant la procédure décrite jusqu'ici peut ne pas être connexe, ce qui n'est pas souhaitable pour des questions de combinatoire. L'utilisation de la triangulation va nous permettre de résoudre ce problème. L'obtention de graphes connexes est une propriété importante pour les algorithmes d'isomorphisme.

4.3 Reconstruction des images

Nous voici désormais en possession d'un graphe. Il nous reste à définir les fonctions P et C introduites dans la partie 4.1, qui nous permettront de reconstruire les images correspondant aux graphes.

P est l'ensemble des coordonnées des pointels d'intérêt. Le passage des coordonnées d'un pixel à celui de ses pointels est représenté sur la figure 4.xiv. Ainsi, si une image a une définition en pixels de $m \times n$, avec des coordonnées allant de $(0,0)$ à $(m-1, n-1)$, alors les pointels ont des coordonnées allant de $(0,0)$ à (m,n).

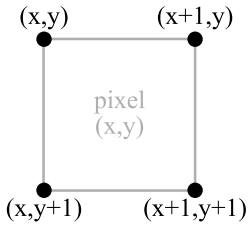


Fig. 4.xiv – Coordonnées : des pixels aux pointels.

Il nous reste à fixer C. Il s'agit des valeurs des couleurs définissant les faces du graphe. Le principe est le suivant : pour chaque face, nous associons une couleur correspondant à la moyenne des pixels la composant, cela sur ses trois composantes RVB. Ceci est simple lorsqu'il s'agit d'une image segmentée (se référer par exemple à la figure 4.xv).

Cependant, lorsque l'on passe aux graphes, un cas particulier est à soulever : celui des pixels traversés par des arêtes. Ces derniers sont traités à part, ils n'interviennent donc pas dans la moyenne d'une face. La couleur de chacun d'entre eux est fixée selon la règle suivante : nous prenons en considération ses quatre pointels et faisons la moyenne de leur couleur. La couleur de chaque pointel correspond elle à la couleur de la région dans laquelle il se trouve. Si un pointel est entre plusieurs régions, sa couleur est alors une moyenne des couleurs des régions auxquelles il appartient. Pour illustrer nos propos, considérons la figure 4.xvi : en haut à gauche, il s'agit d'une partie de la reconstruction d'une image segmentée. Les pixels d'une région ont une couleur correspondant à la moyenne des pixels la composant. Après application de notre méthode d'extraction de graphe, on obtient, par exemple, les arêtes représentées en haut à droite. En bas à gauche : la couleur des faces correspond alors elle aussi à la moyenne des couleurs des pixels la composant (on ignore les pixels traversés par des arêtes). Ainsi, la face bleue

FIG. 4.xv – Reconstruction d'une image segmentée : la couleur d'une région est la moyenne des couleurs de ses pixels.

reste identique, tandis que les couleurs des faces rouge et jaunes sont très légèrement modifiées par la présence, due à l'approximation, d'un pixel d'une autre couleur (respectivement bleu et rouge). Les pixels traversés par des arêtes sont traités à part : on commence par calculer la couleur de leurs pointels, que l'on moyenne (voir le zoom en bas à droite).

En outre, notons que nous choisissons de colorier les faces sans tenir compte des arêtes de la triangulation : en effet, celles-ci ont pour but principal de renforcer la structure des graphes, et n'ont pas de sens sémantique. En toute logique, étant internes aux faces, elles ne modifieraient pas leur couleur, mais pourraient complexifier le calcul inutilement par l'introduction d'autres pixels traversés par des arêtes. Des illustrations de reconstructions de graphes issus de l'image originelle du manchot sont visibles sur la figure 4.xvii. Notons qu'une version de démonstration de notre méthode d'extraction de graphes plans à partir d'images a été développée, et est disponible en ligne à l'URL suivante : `http://labh-curien.univ-st-etienne.fr/EPG/`.

Le graphe permet donc de recoder une certaine version de l'image d'origine, avec moins d'information. Notons, de plus, que l'encodage d'un graphe nécessite une taille inférieure à l'encodage de l'image originelle. En effet, on sait que pour stocker une image de définition $m \times n$ pixels, codée selon un certain modèle de couleur qui correspond à un vecteur de taille K, la taille nécessaire est de $m \times n \times K$ bits (soit $m \times n \times K/8$ octets). Ainsi, pour un codage en trois composantes (par exemple RVB), la taille nécessaire est de $m \times n \times 24$ bits. Or, ici nous avons à coder le graphe plan, c'est-à-dire les coordonnées de ses sommets, ses arêtes, et la couleur des faces. Les coordonnées nécessitent, pour chacun des sommets, une taille de $\log n + \log m$ bits, soit un total de $|V| \times (\log n + \log m)$ bits. Qu'en est-il des arêtes ? Nous avons vu dans la partie 3.1.2 que pour un graphe planaire, l'inégalité suivante était vérifiée : $|E| \leq 3|V| - 6$. On peut donc considérer $3|V|$ comme étant une borne supérieure du nombre de bits nécessaires pour stocker les arêtes. Enfin, intéressons-nous aux couleurs des faces. Une nouvelle fois, la formule d'Euler indique que pour un graphe planaire, le nombre de faces f est tel que : $|V| - |E| + f = 2$. On en déduit que $2|V| \times K$ est une borne supérieure du nombre de bits nécessaires pour

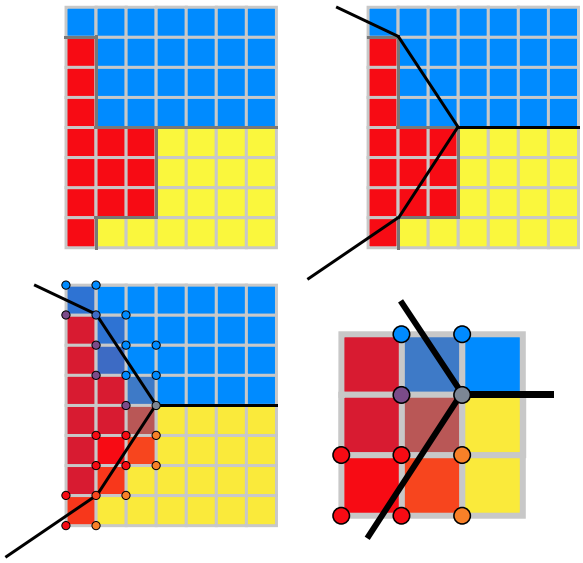


FIG. 4.xvi – Reconstruction des images : calcul de C. En haut à gauche, une partie d'une image segmentée : la couleur d'une région est la moyenne des couleurs de ses pixels. En haut à droite, en noir, le graphe correspondant. En bas à gauche, la valeur de C pour le graphe : les couleurs des faces sont la moyenne des couleurs de leurs pixels, les pixels traversés par des arêtes sont la moyenne des couleurs de leurs pointels. En bas à droite : zoom sur une section de la reconstruction.

FIG. 4.xvii – Reconstruction des images des graphes de la figure 4.xi : 81, 145 et 334 sommets.

stocker les couleurs des faces, sur un vecteur de taille K. Finalement, on obtient une taille d'encodage d'un graphe valant :

$$|V| \times (\log n + \log m + 3 + 2K)$$

Ce chiffre n'est pas une borne supérieure, puisqu'il faut lui ajouter les couleurs des pixels traversés par des arêtes.

4.4 Expérimentations

Afin d'analyser notre méthode d'extraction de graphes plans à partir d'images, nous avons réalisé plusieurs expérimentations. L'intégralité a été menée sur des images provenant de la base d'images segmentées de Berkeley [Martin et al., 2001]. La partie publique de cette base contient 300 images, de définition 481 × 321 pixels, en mode RVB (pour des explications sur ces termes, se référer aux définitions de la partie 1.1.1). Il s'agit de la représentation en couleur de photos mettant en scène des êtres humains, des animaux, des paysages, des constructions... Pour une illustration non exhaustive, consulter la figure 4.xviii. Chacune des images de la base contient, au minimum, un objet discernable. Elles sont en fait toutes extraites de la base d'images COREL[2], largement utilisée dans le domaine de la recherche d'image par le contenu.

FIG. 4.xviii – Exemples d'images provenant de la base de données de Berkeley.

Pour chaque image sont également fournis plusieurs résultats de segmentation, avec plus ou moins de régions, dans des fichiers au format texte. Des représentations sous forme d'images de ces segmentations sont visibles sur la figure 4.xix (il s'agit de segmentation des premières images de chaque ligne des exemples de la figure 4.xviii). Les segmentations ont été faites à la main par des volontaires, sans qu'aucune instruction

[2] http://wang.ist.psu.edu/docs/related/

particulière quant au type de critère à utiliser pour séparer les régions ne leur soit fournie (à l'exclusion du fait que chaque région doive représenter une chose distincte de l'image originelle). Chaque image a été segmentée par plusieurs personnes, à des degrés de finesse différents, selon leur propre perception. Les fichiers contiennent un minimum de deux régions, pour une moyenne de vingt. Pour les 300 images, un total de 1633 segmentations différentes sont disponibles.

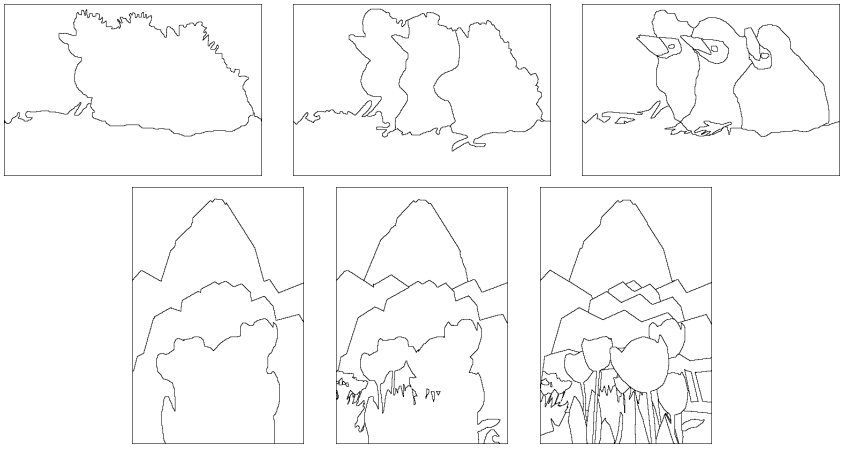


FIG. 4.xix – Exemples d'images montrant des résultats de segmentation et provenant de la base de données de Berkeley.

Quant à l'extraction de pixels d'intérêt, nous avons pour nos expérimentations utilisé la méthode présentée dans [Bres et Jolion, 1999]. Elle part du principe qu'un pixel d'intérêt est caractérisé par une information locale de contraste significative (se référer à la partie 1.1.1 pour la notion de contraste dans une image) et se base sur une analyse multirésolution de l'image (représentée sous la forme d'une pyramide). Celle-ci subit alors une étape de rehaussement de contraste qui vise à réduire l'entropie, et qui engendre une image binaire. Les pixels d'intérêt correspondent alors à des maxima locaux de la mesure de contraste cumulée dans la base de la pyramide.

Nous présentons dans la suite un ensemble d'expérimentations mettant en œuvre la base de données d'images segmentées et l'extracteur de pixels d'intérêt pré-cités. Tout d'abord, il s'agit d'étudier les pertes engendrées par la construction du graphe, par rapport à l'image originelle et à l'image segmentée. Ensuite, on compare ces pertes à une autre méthode d'extraction de graphes. Enfin, on s'intéresse également à l'influence de différents paramètres (nombre de régions, nombre de pixels d'intérêt) sur la taille du graphe obtenu. Nous conserverons les notations introduites dans la partie 4.3, et la fonction *perte* introduite dans 4.1.

4.4.1 Analyse des pertes engendrées par la méthode d'extraction

Nous analysons tout d'abord les pertes dues aux phases de segmentation et de construction du graphe.

Analyse de $perte(I, I_r)$

Nous étudions en premier lieu la perte causée par la segmentation d'une image I en r régions, c'est-à-dire à la valeur de $perte(I, I_r)$. Pour l'ensemble des segmentations disponibles, la perte moyenne est de 0.1004, soit environ 10%. Ceci soulève toutefois une interrogation intéressante : existe-t-il une corrélation entre le nombre de régions de la segmentation d'une image et $perte(I, I_r)$? Pour y répondre, nous avons sélectionné, pour un nombre de régions allant de 3 à 30, 10 résultats de segmentation (soit un total de 280 fichiers), et calculé les pertes moyennes correspondantes. Les résultats sont représentés sur la figure 4.xx, avec les écarts-types correspondants. On remarque clairement qu'il n'existe aucun lien entre le nombre de régions et $perte(I, I_r)$. Ceci peut être expliqué par la façon dont les images ont été segmentées : n'ayant eu aucune instruction particulière pour créer les régions, les personnes ayant segmenté les images n'ont pas nécessairement utilisé des critères liés à des zones de couleur uniformes. Il est même fort probable qu'elles se soient plutôt basé sur des données sémantiques. Il n'y a alors aucune justification pour que ces zones sémantiques soient homogènes, et ce quelque soit la finesse de la segmentation. Les écarts-types, qui sont assez importants quelque soit le nombre de régions, renforcent cette impression.

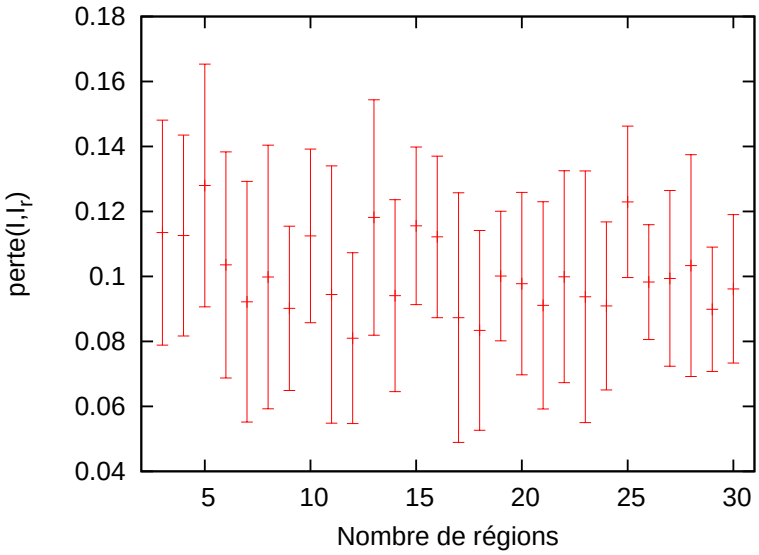


FIG. 4.xx – Valeur de $perte(I, I_r)$ en fonction du nombre de régions des images segmentées, associée aux écarts-types.

Analyse de $perte(I_r, I(G(I_r, p), P, C))$

La seconde perte à laquelle nous nous sommes intéressés est celle qui intervient entre la segmentation et la construction de l'image du graphe, c'est-à-dire $perte(I_r, I(G(I_r, p), P, C))$. Le critère concerne le nombre de pixels d'intérêt extraits p. Pour l'étudier, nous

avons mis en place le protocole suivant : pour toutes les segmentations, nous avons extrait 10 à 500 pixels d'intérêt, ensemble que nous avons affiné selon la démarche décrite dans la partie 4.2.2. Par la suite, les pointels d'intérêt correspondant ont été calculés, et chaque graphe construit. Une fois les images des graphes reconstituées, les valeurs moyennes de $perte(I_r, I(G(I_r, p), P, C))$ ont été obtenues, avec les écarts-types correspondants. Les résultats sont représentés sur la figure 4.xxi : la perte décroît fortement et rapidement lorsque le nombre de pixels d'intérêt extraits augmente. Cela est logique, puisque plus le nombre de pixels est grand, plus les arêtes délimitant les faces suivent les bordures des régions, et donc plus la perte devient faible. Ceci est confirmé par le fait que la valeur des écarts-types diminue nettement avec l'augmentation du nombre de pixels d'intérêt. Enfin, notons que $perte(I_r, I(G(I_r, p), P, C))$ a une valeur moyenne comprise entre 3.12% (pour 10 pixels d'intérêt extraits) et 0.19% (pour 500 pixels d'intérêt extraits), ce qui semble assez faible.

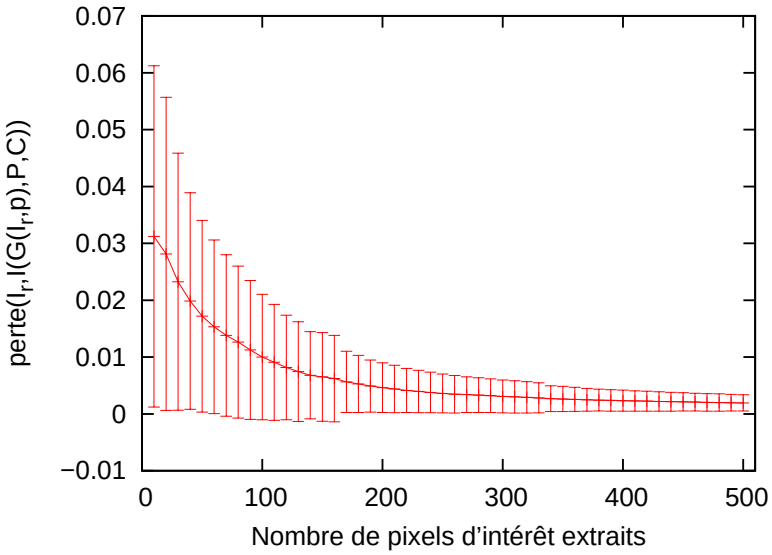


FIG. 4.xxi – Valeur de $perte(I_r, I(G(I_r, p), P, C))$ en fonction du nombre de pixels d'intérêt extraits, associée aux écarts-types.

Analyse de $perte(I, I(G(I_r, p), P, C))$

Pour finir, intéressons-nous à la perte globale engendrée par le processus de construction du graphe à partir de l'image originelle. Celle-ci respecte l'inégalité triangulaire suivante :

$$perte(I, I(G(I_r, p), P, C)) \leq perte(I, I_r) + perte(I_r, I(G(I_r, p), P, C))$$

Sachant cela, il est intéressant d'étudier la part de ces deux étapes dans la perte totale. Nous avons vu qu'en moyenne, $perte(I, I_r) = 10\%$, et que $perte(I_r, I(G(I_r, p), P, C))$

p	r	$\|V\|$	$perte(I, I_r)$	$perte(I_r, I(G(I_r, p), P, C))$	somme
50	3	104	0.084	0.006	0.089
50	6	146	0.083	0.006	0.089
50	7	123	0.084	0.005	0.089
50	9	175	0.084	0.006	0.090
50	13	149	0.084	0.007	0.091
50	18	248	0.076	0.013	0.089
50	20	207	0.083	0.007	0.090

TAB. 4.1 – Pour une image donnée, pertes correspondant à différents niveaux de segmentation.

varie, en fonction du nombre de pixels d'intérêt, de 3% à 0.2% environ. Clairement, la perte la plus importante a toujours lieu lors de la première étape, c'est-à-dire la segmentation de l'image. La méthode que nous utilisons pour ensuite construire un graphe cause largement moins de perte. De plus, elle décroît rapidement pour atteindre 1% dès 100 pixels d'intérêt extraits.

Pour plus de détails, nous avons souhaité étudier le cas d'une image donnée, à différents niveaux de segmentation, cas représenté dans le tableau 4.1. Pour un même nombre de pixels d'intérêt extraits ($p = 50$), et pour un nombre croissant de régions (de 3 à 20), ce tableau apporte en premier lieu des informations sur la taille $|V|$ du graphe obtenu. Comme nous le verrons dans la partie 4.4.3, celle-ci augmente avec le nombre de pixels extraits. Ce sont les trois dernières colonnes qui nous intéressent plus particulièrement. Tout d'abord, elles confirment que la perte a majoritairement lieu lors de la segmentation de l'image. De plus, ces colonnes montrent une nouvelle fois que les deux pertes intermédiaires, et donc la perte totale, sont indépendantes du nombre de régions de l'image.

4.4.2 Comparaison avec une autre méthode d'extraction de graphes

Dans cette partie, nous avons souhaité comparer notre méthode d'extraction de graphes à partir d'images à une autre, plus classique et usitée. Dans cette dernière, il s'agit d'extraire les pixels d'intérêt directement sur l'image originelle (il n'y a donc pas intervention d'une phase préalable de segmentation), auxquels on ajoute les quatre coins de l'image s'ils n'ont pas été détectés. Ces pixels sont alors, par nos critères, convertis en pointels d'intérêt, puis reliés par une triangulation de Delaunay globale. Aucune information sémantique sur les objets figurant sur l'image originelle n'est donc *a priori* conservée. Nous notons $G(I, T_p)$ ce graphe à p pixels d'intérêt extraits, et $I(G(I, T_p), P, C)$ l'image reconstruite (selon les mêmes principes que $I(G(I_r, p), P, C)$. Les pertes engendrées par ces deux méthodes, en fonction du nombre de pixels d'intérêt extraits, sont représentées sur la figure 4.xxii (la valeur de $perte(I, I(G(I_r, p), P, C))$ est obtenue en sommant $perte(I, I_r)$ et $perte(I_r, I(G(I_r, p), P, C))$, ceci est donc une majoration). On remarque que notre méthode induit toujours une perte légèrement

moindre, ce qui la rend compétitive, d'autant plus qu'elle a pour avantage de conserver la sémantique de l'image originelle.

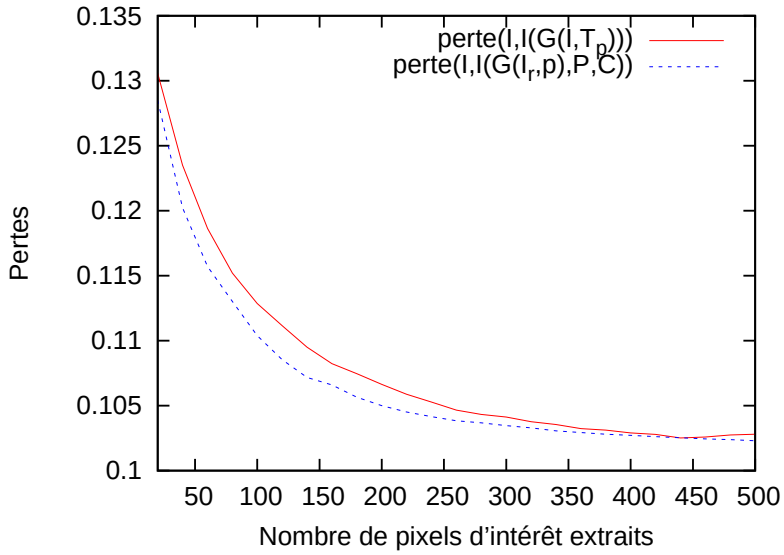


FIG. 4.xxii – $perte(I, I(G(I_r, p), P, C))$ et $perte(I, I(G(I, T_p), P, C))$ en fonction du nombre de pixels d'intérêt extraits.

4.4.3 Influence des paramètres sur la taille du graphe

Intéressons-nous désormais à la taille des graphes obtenus suite à notre méthode de construction.

Comment obtenir un graphe de taille $|V|$?

Étant donné une image, si l'on désire obtenir un graphe d'une taille donnée $|V|$, on peut pour cela se servir des deux paramètres de la méthode : le niveau de segmentation (c'est-à-dire le nombre de régions) et le nombre de pixels d'intérêt extraits. Il est cependant utile de savoir lequel de ces deux paramètres doit être utilisé de préférence : il faut pour cela étudier la perte induite par chacun d'entre eux. Dans ce but, nous avons comparé $perte(I, I(G(I_r, p), P, C))$, pour une même image, et différentes valeurs de r et p menant à un nombre sensiblement égal de pointels d'intérêt. Ainsi, pour deux graphes de même taille issus d'une même image, on peut évaluer s'il est préférable d'avoir un faible nombre de régions et un grand nombre de pixels d'intérêt, ou inversement. Les résultats sont regroupés dans le tableau 4.2 : il s'agit de la situation pour deux images notées A et B. On remarque que la perte totale, $perte(I, I(G(I_r, p), P, C))$, a une valeur assez stable, que l'on agisse sur le nombre de régions ou le nombre de pixels d'intérêt. Cela signifie que chaque paramètre peut être ajusté pour obtenir un graphe d'une taille donnée. Cependant, on peut indiquer une légère préférence pour le cas où le nombre de

Image	p	r	$\|V\|$	$perte(I, I_r)$	$perte(I_r, I(G(I_r, p), P, C))$	$perte(I, I(G(I_r, p), P, C))$
A	80	7	182	0.084	0.004	0.088
	20	18	188	0.076	0.023	0.098
B	80	11	146	0.166	0.009	0.175
	20	25	145	0.166	0.022	0.188

TAB. 4.2 – Influence de r et s sur les pertes lors de l'extraction d'un graphe de taille $|V|$.

pixels d'intérêt est supérieur au nombre de régions. Effectivement, dans cette configuration, $perte(I_r, I(G(I_r, p), P, C))$ diminue puisque, comme nous l'avons déjà expliqué dans la partie 4.4.1, les arêtes représentent alors de façon plus juste les bordures de segmentation. On pourrait estimer que pour extraire un graphe de taille approchant $|V|$, il faudrait fixer $r = |V|/20$ et $p = |V|/2$.

Relation entre le nombre de pixels d'intérêt extraits et $|V|$.

Soyons plus précis. Nous souhaitons mettre en relation le nombre de pixels d'intérêt extraits et le nombre de pointels obtenus afin d'évaluer clairement l'influence de la valeur de p. Dans ce but, nous avons extrait 10 à 500 pixels d'intérêt sur les 1633 images de segmentation disponibles, nous avons affiné cet ensemble comme décrit dans la partie 4.2.2 et calculé le nombre de pointels d'intérêt correspondant. Les résultats sont visibles sur la figure 4.xxiii : $|V|$ est quasi-linéairement dépendant de p, et on pourrait approximer cette courbe par une droite d'équation $y = 1.4 \times x + 135$.

Relation entre le nombre de régions et $|V|$.

Pour finir, nous focalisons notre attention sur l'influence de r sur la taille du graphe. Pour cela, nous avons sélectionné, pour un nombre de régions allant de 3 à 30, 10 images de segmentation (soit un total de 280 fichiers). Pour chacune, nous avons extrait 50, 100, 150 et 200 pixels d'intérêt, qui ont été affinés puis transformés en pointels d'intérêt. Les résultats sont représentés sur la figure 4.xxiv : bien que la croissance des courbes ne soit pas régulière, on remarque que, pour une valeur p donnée, plus le nombre de régions augmente et plus le nombre de pointels d'intérêt augmente lui aussi. Ceci est en majorité lié à l'augmentation du nombre de pixels d'intersection de régions, qui sont ajoutés lors de la phase d'affinage (partie 4.2.2).

4.5 Discussion

Nous avons présenté une méthode d'extraction de graphes plans à partir d'images, qui nous permet d'obtenir ce que nous jugeons être de "bons" graphes pour des tâches

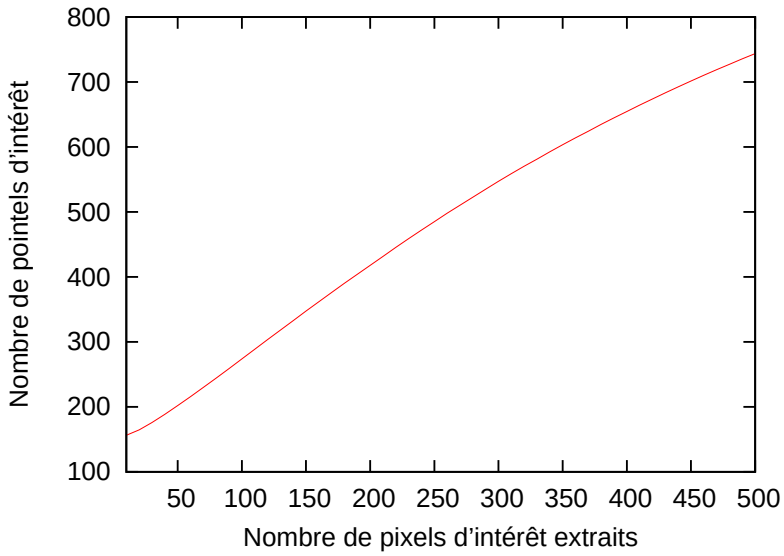


FIG. 4.xxiii – Des pixels d'intérêt aux pointels d'intérêt.

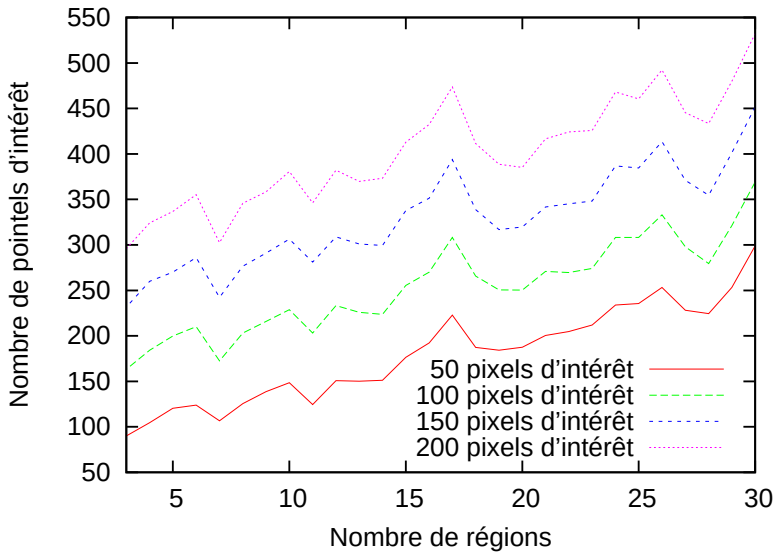


FIG. 4.xxiv – Influence du nombre de régions sur le nombre de pointels d'intérêt, en fonction du nombre de pixels d'intérêt extraits.

de reconnaissance de formes : ils sont à la fois intéressants algorithmiquement parlant, de par leur taille modérée et le fait qu'ils soient connexes et plans ; mais ils ont également une dimension sémantique qui caractérise en grande partie l'originalité de notre méthode et son intérêt pour des domaines d'applications liés à l'image. Cette méthode permet également de reconstruire les images associées aux graphes, avec une perte d'information. Les expérimentations ont montré que les pertes engendrées étaient limitées.

Plusieurs pistes de travail intéressantes pourraient être explorées. En premier lieu, d'autres fonctions de pertes pourraient être proposées : la distance L1 sur le codage RVB des pixels n'est peut-être pas la plus adaptée pour juger d'une perte visuelle. Ainsi, la mesure de distorsion PSNR (*Peak Signal to Noise Ratio*) est plus fréquemment utilisée, notamment en compression d'images. Elle est définie par : $PSNR = 10.\log_{10}\left(\frac{d^2}{EQM}\right)$, où d est la plus grande valeur d'un pixel (ce qui correspond à 255 s'il est codé sur 8 bits). EQM est l'erreur quadratique moyenne, définie pour deux images I_1 et I_2 de taille $m \times n$ comme suit : $EQM = \frac{1}{mn}\sum_{i=0}^{m-1}\sum_{j=0}^{n-1}||I_1(i,j) - I_2(i,j)||^2$. On peut également citer la mesure de similarité entre images SSIM, qui a été développée en s'intéressant plus particulièrement à la différence de structure entre les deux images [Wang et al., 2004].

En outre, les techniques de simplification polygonales évoquées à la fin de la partie 4.2.1 pourraient être utilisées pour simplifier les contours des régions, à la place des détecteurs de points d'intérêt, dont l'utilisation est peu naturelle sur des images binaires. Citons également les travaux sur les *alpha shapes* [Stelldinger et al., 2006].

De plus, la phase d'affinage visant à compléter l'ensemble des pixels d'intérêt, avant de les transformer en pointels, pourrait avoir lieu directement sur les pointels, afin d'éviter de détecter de multiples pixels d'intérêt menant au même pointel.

D'autre part, la triangulation de Delaunay n'est pas la seule possibilité pour donner plus de robustesse au graphe et le rendre connexe. Il est par exemple possible d'ajouter des arêtes aux graphes en associant plusieurs opérations de traitement d'images, comme illustré sur la figure 4.xxv. La première phase consiste, classiquement, en la détection des bordures des régions (image (a), ici obtenues par dilatation). Ensuite, il s'agit de réaliser la carte géodésique des distances aux bords pour chaque région : on calcule la distance de chacun des pixels par rapport aux bords de la région à laquelle il appartient. Ainsi, plus l'on s'éloigne du bord et plus la distance augmente (graphiquement, on se rapproche donc de plus en plus d'une teinte claire, comme sur l'image (b)). Cette carte géodésique est ensuite inversée (image (c)) pour permettre le calcul de la dernière étape : la ligne de partage des eaux. Il s'agit de considérer l'image de la carte géodésique comme une surface en trois dimensions, hauteur, largeur et niveau de gris. Ensuite, on peut imaginer que l'on fait monter un niveau d'eau, en partant du niveau de gris 0. Ainsi, on va peu à peu remplir des "bassins" (ou des cuvettes). Avec la montée de l'eau, lorsque deux bassins vont se rencontrer pour n'en former plus qu'un, on conserve la ligne de démarcation, qui forme donc un segment à l'intérieur de la région. L'image (d) illustre cet ajout de segments : le contour est désormais connexe. Cependant, certains segments ajoutés ne sont pas forcément pertinents, comme on peut le voir à gauche du cou du manchot. Ils sont dus à l'existence de petits bassins rapprochés, sortes de minima locaux. Pour pallier ce problème, on peut choisir de précéder la phase de détection des lignes de partage des

eaux par une suppression de ces minima locaux (en fait, il s'agit de combler les bassins de hauteur inférieure à un seuil donné). Sur l'image (e), on a choisit de combler tout bassin de hauteur inférieure à 1. Ceci a permis de supprimer plusieurs segments qui n'étaient pas forcément pertinents. Cependant, nous avons perdu la connexité. Il est donc une nouvelle fois question de compromis, en sachant qu'une solution unique pour toutes les images n'existe pas.

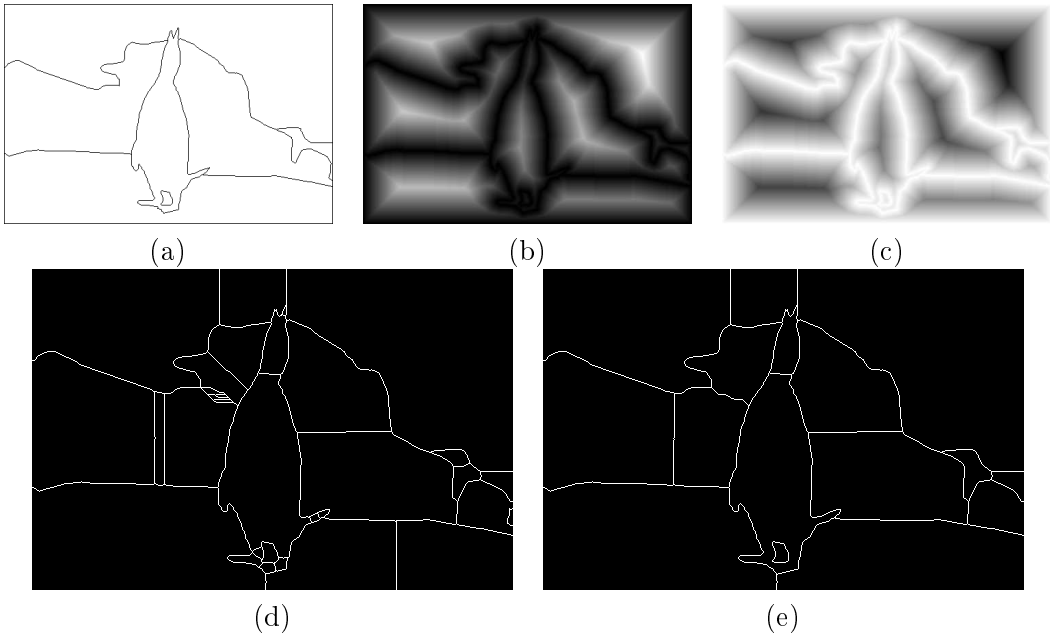


FIG. 4.xxv – Ajout de robustesse aux graphes : une autre approche.

En outre, les expérimentations pourraient être complétées par des comparaisons utilisant différents algorithmes de segmentation (bien que le but ne soit pas d'évaluer ces algorithmes). En effet, les segmentations utilisées proviennent d'une base de donnée constituée manuellement : la qualité de la segmentation est donc élevée, sémantiquement parlant. Or, il serait bien difficile d'obtenir les mêmes résultats avec des algorithmes automatiques. Une comparaison des pertes actuelles avec celles engendrées par de algorithmes automatiques de segmentation serait judicieuse. D'autre part, nous pourrions également comparer notre extraction à d'autres méthodes de construction de graphes, comme le RAG.

Finalement, il serait évidemment intéressant d'utiliser ces graphes en reconnaissance de formes, pour évaluer leur intérêt.

5

Algorithmes polynomiaux d'isomorphismes de graphes plans à trous

Sommaire

Ce chapitre est consacré à l'étude de graphes plans particuliers, qui résultent de la représentation d'images sous forme de graphes et de la recherche de motifs. Il s'agit des *graphes plans à trous*, qui sont caractérisés avant tout par leurs faces. Nous reviendrons pour cela sur les plongements des graphes planaires. Puis nous définirons différentes notions d'isomorphismes de graphes plans à trous, avant de mettre en avant le fait qu'il peut exister des graphes qui, bien que non isomorphes, sont *équivalents*, notion que nous approfondirons. Nous décrirons alors un processus permettant, étant donné un graphe plan à trous, d'obtenir une forme irréductible équivalente : la *normalisation*. Par la suite, nous donnerons plusieurs algorithmes permettant de résoudre isomorphismes et équivalence en temps polynomial. Nous nous intéresserons également à des sous-graphes particuliers et définirons pour cela la notion de *motif*. Nous verrons que rechercher un motif dans un graphe reste un problème polynomial, et présenterons quelques expérimentations portant sur la recherche de motifs dans des images. Enfin, nous ouvrirons sur certaines pistes concernant la connexité des graphes considérés.

5.1 Recherche de motifs

Nous l'avons déjà évoqué : lorsque les graphes représentent des images, ils ont pour caractéristiques d'être planaires, et surtout plans. C'est-à-dire que leur plongement est donné, et qu'ils sont dessinés de façon à ce qu'aucune de leurs arêtes ne se rencontre, sauf aux extrémités. Ceci signifie que visuellement, on peut considérer un tel graphe comme étant un ensemble de faces ayant un agencement donné.

Les applications en reconnaissance de formes ou en classification d'images s'attachent à rechercher des graphes ou des sous-graphes. En fait, il s'agit souvent de retrouver des ensembles de faces. C'est ce que nous appellerons des *motifs*. Lors de ces recherches, deux cas sont envisageables.

Motifs compacts

Dans le cas général, qui peut être qualifié de simple, le motif à trouver peut être qualifié de *compact* : il correspond à un ensemble de faces, qui sont toutes recherchées. C'est par exemple le cas représenté sur la figure 5.i : si l'on recherche dans une autre image le sous-graphe correspondant à la fleur, ou à l'un de ses boutons, on souhaite retrouver toutes les faces de ce sous-graphe.

Motifs à trous

Cependant, les motifs recherchés peuvent être plus complexes. Sur la même image, on peut vouloir retrouver ensemble les parties du graphe correspondant à la fleur et aux boutons, mais pas celles correspondant au fond. Dans ce cas, on cherche certaines faces, mais pas d'autres : il s'agit d'un motif *à trous*. Le graphe qui le dénote est, lui aussi, *à trous*, c'est-à-dire que l'on peut distinguer deux types de faces : les faces recherchées qui sont *visibles* et les autres, qui sont *invisibles*. Ce sont ces dernières qui constituent les trous.

FIG. 5.i – Cas général d'un graphe issu d'une image : le motif est compact.

Le cas des motifs à trous est en fait fréquent et assez intuitif, dans le domaine de l'image. En effet, beaucoup de motifs, qui ne représentent pourtant qu'un seul objet, peuvent être à trous. Ceci se vérifie par exemple pour l'image 5.ii. Si l'on souhaite retrouver le graphe correspondant au *mug* uniquement, alors on voit clairement qu'il est à trous : toutes les faces de l'intérieur de l'anse sont invisibles. Le *mug* doit pouvoir être localisable dans une autre image, indépendamment des changements qui peuvent avoir lieu dans l'anse (pour cause de changement de fond, par exemple).

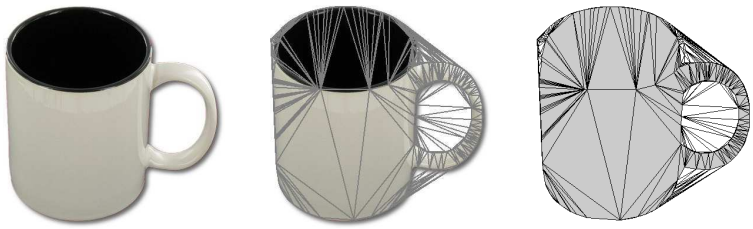


FIG. 5.ii – Le graphe correspondant au *mug* est à trous : les faces grises sont visibles, mais celles de l'intérieur de l'anse, en blanc, sont invisibles : on ne souhaite pas les prendre en compte pour la recherche du *mug* dans des images.

Outre le domaine de l'image, la notion de faces visibles et invisibles peut également s'appliquer à la modélisation de l'environnement de robots. Ceux-ci ont effectivement besoin de cartes pour savoir comment et où se déplacer dans l'espace. Ces cartes sont évidemment modélisables par des graphes [Choset, 2005] : si l'on considère par exemple un appartement, les sommets peuvent alors être les angles des pièces, et les arêtes leurs murs (d'autres types de cartes sont évidemment possibles). Ceci permet de conserver à la fois la forme des pièces et la façon dont elles sont agencées les unes par rapport aux autres : c'est la géométrie et la topologie de l'appartement. Or, il est possible que l'on souhaite empêcher le robot de pénétrer dans certaines pièces de l'appartement. En terme de graphe à trous, ceci revient à définir des faces visibles, qui seront les pièces accessibles par le robot, et des faces invisibles, étant celles qui lui sont interdites. Un

exemple est représenté sur la figure 5.iii.

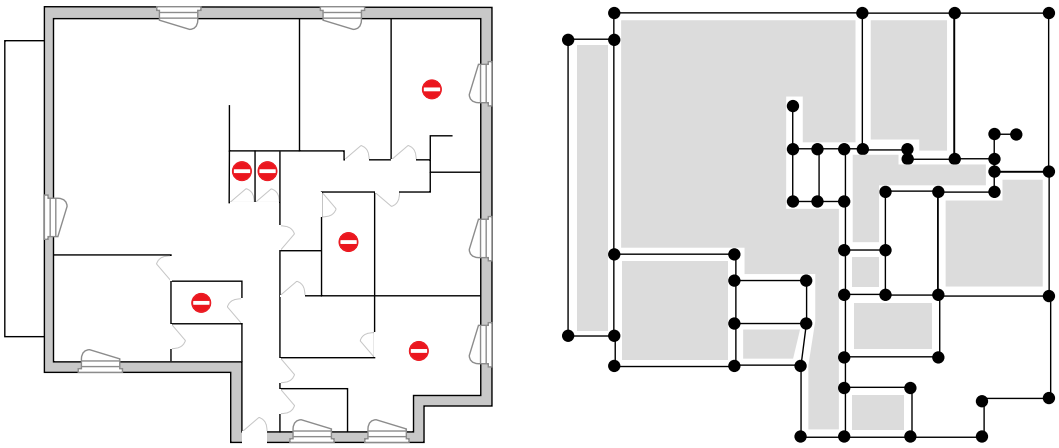


FIG. 5.iii – Modélisation d'un appartement sous forme de graphe pour un robot : les pièces interdites d'accès sont les faces invisibles (en blanc) du graphe à trous.

Nous reviendrons largement sur ces notions dans la suite de ce chapitre, et nous nous intéresserons particulièrement à la recherche des motifs à trous dans des graphes.

5.2 Retour sur les graphes plans

Un graphe plan est une représentation plane d'un graphe planaire (la partie 3.1.2 comporte les définitions nécessaires). Dans cette partie, nous reviendrons plus en détails sur les différentes représentations planes d'un même graphe planaire. Puis nous introduirons plusieurs définitions et notations nous permettant de caractériser et manipuler les graphes plans à trous.

5.2.1 Graphes planaires, plongements et isotopies

Un graphe planaire admet une infinité de représentations planes : il suffit pour cela de modifier légèrement les coordonnées de ses sommets pour en obtenir une nouvelle. Mais le nombre de représentations planes redevient fini si les plongements considérés sont *isotopes* [Fusy, 2007], c'est-à-dire si l'on peut déformer l'un pour obtenir l'autre, de façon continue, sans qu'aucun croisement d'arêtes n'ait lieu. Ainsi, les deux plongements du graphe planaire de la figure 5.iv sont isotopes, ce qui n'est pas le cas des plongements du graphe planaire de la figure 5.v : pour obtenir l'un d'eux à partir d'un autre en déplaçant les sommets, il faut obligatoirement que des intersections entre arêtes aient lieu à un moment donné.

À ce stade, il est particulièrement important de distinguer les notions d'isomorphismes, qui concernent les graphes en tant qu'objets mathématiques, et les isotopies, qui portent sur les plongements (dessins) de ces graphes. On peut par exemple observer

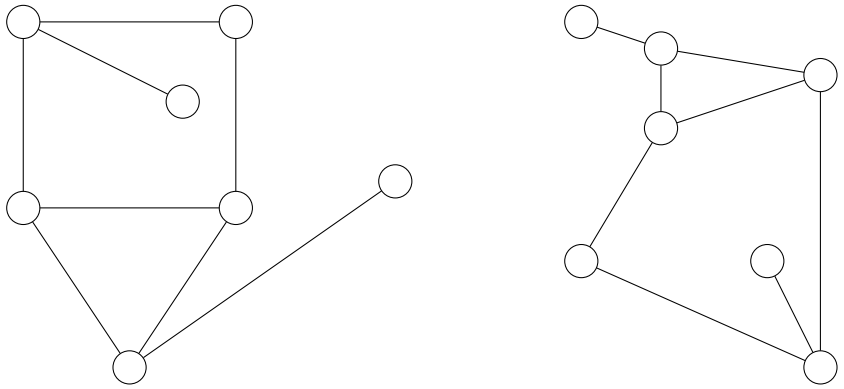


FIG. 5.iv – Deux plongements isotopes d'un même graphe planaire.

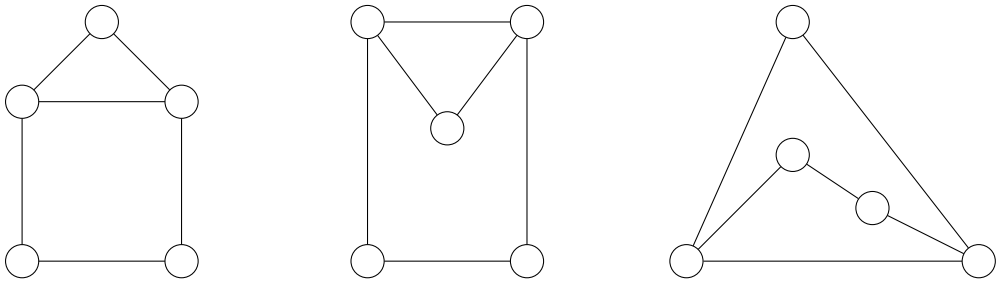


FIG. 5.v – Trois plongements non isotopes d'un même graphe planaire.

que la notion d'isotopie garde un sens quelque soit la surface sur laquelle on réalise le plongement (un plan, une sphère, un tore...), mais que la comparaison de plongements sur des surfaces différentes n'a pas de sens. À l'inverse, la définition habituelle de l'isomorphisme de graphes ne tient pas compte du dessin. Ainsi, les graphes de la figure 5.v sont isomorphes dans le sens strict de la définition, mais leurs plongements ne sont pas isotopes.

Or, si ces plongements de graphes sont des représentations d'images, alors ils ne reproduisent assurément pas les mêmes objets. Au contraire, les plongements du graphe planaire de la figure 5.iv appartiennent à la même classe d'isotopie, et visuellement il est effectivement plus probable qu'ils soient issus d'images semblables à quelques déformations près (changements d'échelle, rotations ou autres déformations limitées). Ainsi, c'est bien la notion d'isotopie de plongements de graphes planaires qui nous intéresse en pratique.

Pourtant, nous aspirons à travailler sur les graphes eux-mêmes. Aussi, en premier lieu, nous allons enrichir la définition des graphes planaires, en y intégrant des informations sur les faces, permettant de capturer syntaxiquement toute une classe d'isotopie de ses plongements ; nous parlerons alors de *système de description d'un graphe plan (SDGP)* (voir la partie 5.2.3). En second lieu, nous définirons l'*isomorphisme plan*, étendant la notion habituelle d'isomorphisme avec des contraintes de préservation des faces (voir la partie 5.3).

Nous avons déjà fait remarquer, dans la partie 3.1.2, que pour un graphe planaire connexe $G = (V, E)$, ayant $|F|$ faces, l'égalité suivante se vérifie

$$|V| - |E| + |F| = 2.$$

Le nombre de faces d'un graphe planaire donné est donc invariant, quelque soit la classe d'isotopie de son plongement. Ce n'est donc pas le nombre de faces qui permet de distinguer une classe d'isotopie d'une autre, mais bien les faces elles-mêmes. Ainsi, les plongements de la figure 5.vi ne sont pas les mêmes aux isotopies près, et cela se vérifie de par la configuration de leurs 3 faces : celui de gauche est formé de cycles de longueur 4 (f_2), 5 (f_3, les arêtes pendantes participant deux fois lors du parcours de la face) et 5 (f_1, face externe), tandis que celui de droite est composé de cycles de longueur 4 (f_2), 3 (f_3) et 7 (f_1, face externe).

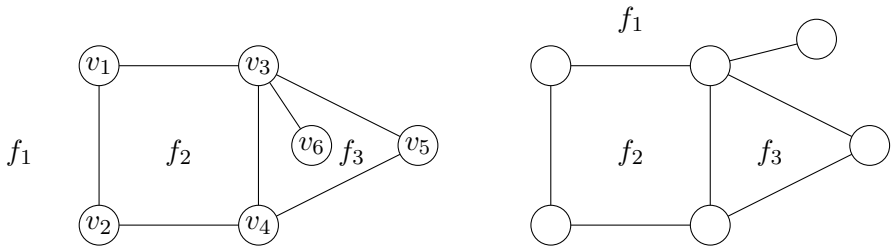


FIG. 5.vi – Deux plongements d'un même graphe planaire, qui diffèrent par les faces : cycles de longueur 4 (f_2), 5 (f_3) et 5 (f_1) pour celui de gauche, et 4 (f_2), 3 (f_3) et 7 (f_1) pour celui de droite.

Cependant, les cycles seuls ne suffisent pas toujours à distinguer une classe d'isotopie d'une autre. Ainsi, les plongements des graphes de la figure 5.v ne peuvent pas être différenciés uniquement par les contours des faces. En fait, la seule manière de les différencier est de distinguer la face externe des autres faces.

Par conséquent, cette vision des graphes plans nous amène à les définir non plus comme de simples plongements de graphes planaires, mais bien comme un modèle symbolique caractérisé avant tout par ses faces.

5.2.2 Notions de connexité et de faces contiguës

Dans la suite de ce chapitre, nous poserons sur les graphes étudiés des contraintes de connexité. En effet, nous verrons dans la partie 5.7.2 que nous ne savons pas étendre entièrement nos résultats aux graphes planaires non connexes pour l'instant.

Définition 5.1 (Graphe connexe)
Un graphe $G = (V, E)$ est connexe *si, pour tout couple de sommets $(v, v') \in V$, il existe*

une séquence de sommets $v = v_0, v_1, \ldots, v_n = v' \in V$ *telle que,* $\forall i \in \{0, 1, \ldots, n-1\}$, $\{v_i, v_{i+1}\} \in E$.

Ainsi, le graphe de la figure 5.vii n'est pas connexe : le sommet 5 est isolé (son degré vaut 0), le 6 et le 7, bien que reliés entre eux, ne sont pas accessibles depuis les autres sommets du graphe.

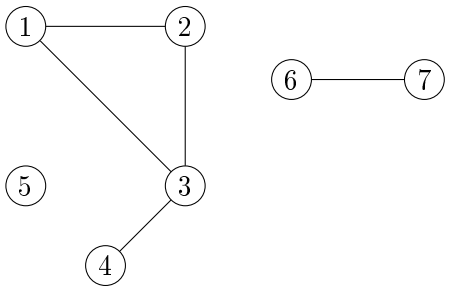


FIG. 5.vii – Un graphe non connexe.

Cependant, si l'on en revient à l'analogie des robots, le fait qu'un graphe soit connexe ne suffit pas à lui permettre de se déplacer dans toutes les faces autorisées, quel que soit son point de départ. À cet effet, considérons le graphe de la figure 5.viii. Si le robot a la possibilité de se déplacer dans l'ensemble de faces $\{f_1, f_3, f_5\}$, alors, quelle que soit la face de départ, il ne pourra atteindre aucune des deux autres puisque ces pièces ne communiquent pas entre elles. Nous dirons que cet ensemble n'est pas constitué de faces *contiguës*. À l'inverse, si l'ensemble est $\{f_1, f_4, f_5\}$, alors tous les déplacements sont possibles et les faces sont contiguës.

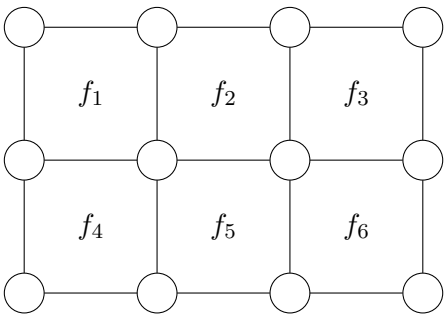


FIG. 5.viii – $\{f_1, f_3, f_5\}$ n'est pas un ensemble de faces contiguës, à l'inverse de $\{f_1, f_4, f_5\}$.

Définition 5.2 (Ensemble de faces contiguës)
Soit K *un ensemble de faces. Les faces de* K *sont* contiguës *si, pour tout couple de*

faces $f, f' \in K$*, il existe une séquence de faces* $f = f_0, f_1, \ldots, f_n = f' \in K$ *telle que,* $\forall i \in \{0, 1, \ldots, n-1\}$*,* f_i *et* f_{i+1} *ont au moins une arête commune.*

5.2.3 Système de description d'un graphe plan

Introduisons tout d'abord quelques notations permettant de caractériser et manipuler les graphes planaires connexes que nous allons définir. Il s'agit surtout de dénoter les faces.

Soit V un ensemble fini non vide de symboles. Un *mot* u sur V est une concaténation de tels symboles. L'ensemble des mots possibles est dénoté par V^*, et le mot vide par ε. Un *mot circulaire* $[u]$ est un mot dont le dernier symbole est suivi par le premier : en réalité il n'y a donc pas de notion de premier ou dernier symbole, mais une fonction associant à tout symbole le symbole suivant. Ainsi, si u et v sont deux mots, alors $[uv] = [vu]$. L'ensemble des mots circulaires possibles sur V sera noté V^C. Chaque face d'un graphe plan peut désormais être désignée par un mot circulaire sur ses sommets, tel que deux sommets consécutifs soient reliés par une arête, ainsi que le dernier et le premier sommet (rappelons que nous ne manipulons que des graphes connexes : un seul mot circulaire est donc suffisant pour dénoter une face).

Il nous reste à définir le sens d'un cheminement le long d'une face. Par convention, nous parcourrons une face en laissant ce qui ne lui appartient pas sur notre droite (comme si nous marchions le long de ses arêtes, avec l'intérieur de la face à notre gauche : on peut visualiser une face comme étant un lac possédant des pontons qui forment les sommets, et que l'on parcourrait dans un sens donné). De ce fait, en prenant pour exemple le graphe plan de la gauche de la figure 5.vi, la face f_2 a pour frontière le mot circulaire $[v_2v_4v_3v_1]$, ou, de façon équivalente, $[v_3v_1v_2v_4]$, la frontière de f_3 est $[v_5v_3v_6v_3v_4]$ et celle de f_1 est $[v_1v_3v_5v_4v_2]$.

Nous pouvons désormais définir le système de description d'un graphe plan :

Définition 5.3 (Système de description d'un graphe plan)
Un système de description d'un graphe plan *(SDGP) est un quadruplet* $G = (V, F, e, \mathcal{D})$ *tel que :*

- V *est un ensemble fini non vide de symboles appelés sommets ;*
- F *est un ensemble fini non vide de symboles appelés faces ;*
- $e \in F$ *est la* face externe *;*
- $\mathcal{D} : F \to V^C$ *est une fonction de description des frontières des faces, qui à toute face associe son mot circulaire.*

Nous noterons $\mathbf{f}$ la valeur de $\mathcal{D}(f)$ pour plus de clarté. Nous pourrons alors laisser la fonction $\mathcal{D}$ implicite, et dénoter un SDGP par le triplet (V, F, e). Illustrons notre définition avec le graphe de gauche de la figure 5.vi : il correspond au triplet (V, F, e) avec $V = \{v_1, v_2, \ldots, v_6\}$, $F = \{f_1, f_2, f_3\}$, $e = f_1$ et

$$\begin{aligned} \mathbf{f_1} &= [v_1v_3v_5v_4v_2], \\ \mathbf{f_2} &= [v_1v_2v_4v_3], \\ \mathbf{f_3} &= [v_4v_5v_3v_6v_3]. \end{aligned}$$

Tout graphe plan connexe peut être décrit sous la forme d'un SDGP. Le contraire n'est pas forcément vrai : tout SDGP ne décrit pas forcément un graphe plan connexe. Il faut pour cela que certaines conditions supplémentaires soient respectées.

Pour les énoncer, nous commençons par introduire les notions d'arcs et arêtes. Celles-ci découlent du parcours des faces : ce dernier se fait en laissant ce qui ne leur appartient pas sur la droite. Ceci signifie que lorsque l'on décrit toutes les faces d'un graphe, toute arête $\{x, y\} \in E$ intervient deux fois, le sens du trajet le long de l'arête étant inversé. En fait, tout se passe comme si chaque arête était décomposée en deux arcs orientés (voir la figure 5.ix).

FIG. 5.ix – Notions d'arêtes et d'arcs : chaque couleur désigne la frontière d'une face, dont le parcours s'effectue dans le sens des flèches.

Définition 5.4 (Arcs et arêtes)
Soit $G = (V, F, e)$ un SDGP. On définit :
- *l'ensemble A des* arcs, *avec $A = \{xy \in V^2 \ : \ \exists f \in F, \ \exists u \in V^* \mid \mathbf{f} = [xyu]\}$;*
- *l'ensemble E des* arêtes, *avec $E = \{\{x, y\} \mid xy \in A \text{ et } yx \in A\}$.*

Disposant d'une définition formelle des arêtes dans le cadre des SDGP, on peut maintenant redéfinir les notions de faces adjacentes et d'ensembles de faces contiguës de la façon suivante :

Propriété 5.1 (Ensemble de faces contiguës)
Soit $G = (V, F, e)$ un SDGP.
- *deux faces f et f' sont* adjacentes *si $\mathbf{f} = [xyu]$ et $\mathbf{f'} = [yxv]$ avec $x, y \in V$ et $u, v \in V^*$;*
- *les faces d'un ensemble $K \subseteq F$ sont dites* contiguës *si, pour tout couple de faces $f, f' \in K$, il existe une séquence de faces $f = f_0, f_1, \ldots, f_n = f' \in K$, telle que $\forall i \in \{0, 1, \ldots, n-1\}$, f_i et f_{i+1} sont adjacentes.*

Nous pouvons désormais définir un SDGP bien fondé :

Définition 5.5 (Système de description d'un graphe plan bien fondé)
Un système de description d'un graphe plan $G = (V, F, e)$ est dit bien fondé *(SDGP bien fondé) si les conditions suivantes sont vérifiées :*

1. *tout arc intervient dans une face et une seule :* $\forall x, y \in V,\ f, g, \in F,\ u, v \in V^*$,

$$\mathbf{f} = [xyu] \text{ et } \mathbf{g} = [xyv] \Longrightarrow u = v \text{ et } f = g$$

2. *à tout arc correspond une arête du graphe plan associé :* $\forall x, y \in V$,

$$xy \in A \Longrightarrow \{x, y\} \in E,$$

   ou, de façon équivalente, $(xy \in A \Longrightarrow yx \in A)$;

3. F *est un ensemble de faces contiguës (au sens de la définition 5.1) ;*
4. *tout sommet concourt à la description d'au moins une face :* $\forall v \in V$, $\exists f \in F$ *telle que* $\mathbf{f} = [vu]$ *pour un certain* $u \in V^*$.

Ce qui nous mène au résultat suivant :

Conjecture Soit $G = (V, F, e)$ un SDGP. Si G est bien fondé, alors il existe un unique plongement, aux isotopies près, tel que $G' = (V, E)$ soit un graphe plan connexe formé des faces de F et ayant pour face externe e.

Le principe de l'algorithme construisant le plongement du graphe est le suivant : la première étape consiste à dessiner la frontière de la face externe e. Ensuite, il s'agit de choisir un des arcs xy de cette dernière, puis de sélectionner l'arc yx opposé. Celui-ci existe (condition *2.*) et correspond à la frontière d'une face qui existe également et qui est unique (condition *1.*). On peut alors dessiner cette face à l'intérieur de la face externe, en fusionnant les arcs opposés. On répète le processus, en sélectionnant à chaque fois un arc déjà dessiné, et dont l'arc opposé n'a pas encore été considéré, jusqu'à ce que toutes les faces soient dessinées. La condition *3.* assure que tous les éléments de F sont bien sur le plongement puisqu'accessibles par un chemin de faces depuis la face externe. De plus, par la condition *4.*, tous les sommets de V sont également dessinés. Tout au long du processus, il faut également veiller à ce qu'aucune des arêtes ne se croise : il peut être alors nécessaire d'utiliser des courbes plutôt que des segments de droites. La dernière étape de l'algorithme peut consister à déplacer les sommets afin d'obtenir uniquement des segments (conformément au théorème évoqué dans la partie 3.1.2 [Fáry, 1948]).

5.2.4 Graphes plans à trous

Nous l'avons rappelé dans la première partie de ce chapitre : lorsque les graphes représentent des images (ou encore l'environnement de robots), certaines faces peuvent faire partie du fond, ou même être occultées (une partie de l'objet recherché peut être dissimulée). C'est la raison pour laquelle nous introduisons les graphes plans à trous, pour lesquels chaque face se trouve être visible ou invisible.

Définition 5.6 (Graphe plan à trous)
Un graphe plan à trous *est un quintuplet* $G = (V, E, F, F_v, e)$ *tel que :*
- (V, F, e) *est un SDGP bien fondé, comme caractérisé par la définition 5.5 ;*
- E *est l'ensemble des arêtes (voir la définition 5.4) ;*

- $F_v \subseteq F$ *est un sous-ensemble de faces contiguës, qui sont appelées faces* visibles *;*
- *toute face appartenant à* $F \backslash F_v$ *sera désignée comme étant* invisible *;*
- e *est la face externe.*

Notons que l'ensemble des arêtes E pourrait rester implicite, comme c'est le cas dans la définition d'un SDGP, et être déduit des faces. De plus, remarquons que $F \backslash F_v$ n'est pas forcément connexe ni formé de faces contiguës (le graphe pris dans sa globalité est bien, lui, connexe). Ainsi, le graphe de l'image 5.x est un graphe plan à trous, pour lequel les faces visibles sont grisées (ce sera toujours le cas dans les illustrations de ce chapitre). Ces dernières sont contiguës. Les faces invisibles n'ont elles pas de contrainte de connexité particulière. La face externe peut être visible ou invisible, nous y reviendrons. La figure 5.xi montre, elle, une sélection de faces visibles qui ne conduit pas à un graphe plan à trous.

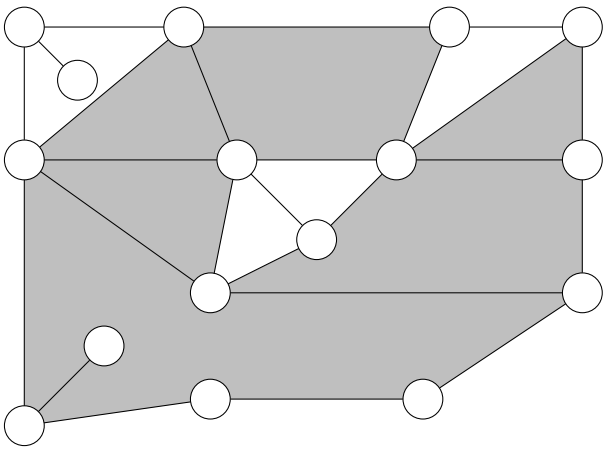


Fig. 5.x – Exemple de graphe plan à trous.

5.2.5 Notations complémentaires

Nous terminons cette partie par quelques notations. Considérons un arc $xy \in A$. Nous notons $\underrightarrow{xy}$ la face unique f telle que $\mathbf{f} = [xyu]$ pour un certain $u \in V^*$. De plus, étant donné une arête $a = \{x, y\} \in E$, nous dénoterons par $faces(a)$ l'ensemble des faces incidentes à a, c'est-à-dire : $faces(\{x, y\}) = \{\underrightarrow{xy}, \underrightarrow{yx}\}$. Notons que $faces(a)$ peut ne contenir qu'une seule face si $\underrightarrow{xy} = \underrightarrow{yx}$. De plus, étant donné un sommet $v \in V$, nous définissons le voisinage de v comme étant un mot circulaire, noté $\Gamma(v)$, formé des sommets adjacents à v, rencontrés dans le sens inverse des aiguilles d'une montre. $|\Gamma(v)|$ correspond ainsi au degré de v. Pour illustrer ces notations, considérons la figure 5.xii. Soient les arcs xy en rouge, yx en vert, yz en bleu et zy en jaune. Alors, $\underrightarrow{xy} = f_3$, $\underrightarrow{yx} = f_1$,

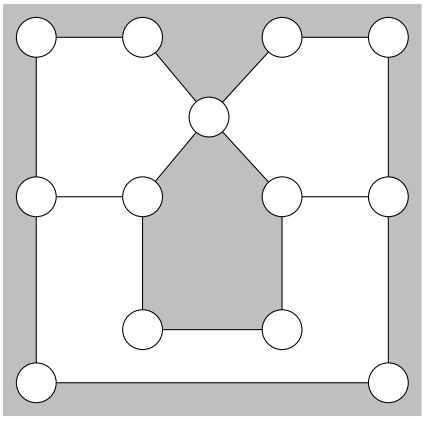


FIG. 5.xi – Une sélection de faces visibles qui n'est pas contiguë et ne correspond donc pas à un graphe plan à trous.

$\overrightarrow{yz} = f_3$ et $\overrightarrow{zy} = f_3$. On a également $faces(\{x, y\}) = \{f_3, f_1\}$, et $faces(\{y, z\}) = \{f_3\}$. Enfin, $\Gamma(y) = [xvwz]$ et $|\Gamma(y)| = 4$.

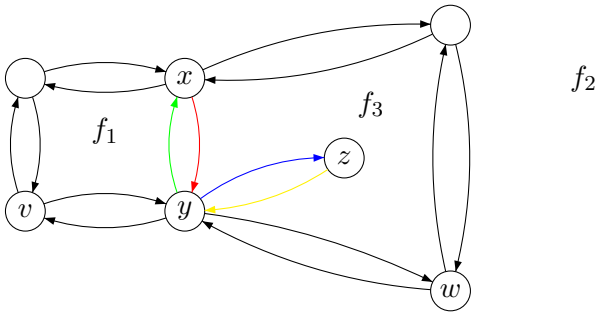


FIG. 5.xii – Illustration des notations.

5.3 Isomorphisme de graphes plans à trous

Les graphes à trous que nous venons de définir, et leur intérêt dans le domaine de l'image, nous conduisent à distinguer et définir plusieurs types d'isomorphismes. Nous reviendrons brièvement sur l'isomorphisme classique, avant de nous focaliser plus particulièrement sur de nouvelles notions : l'isomorphisme sphérique et l'isomorphisme plan.

5.3.1 Isomorphisme classique

Commençons donc par rappeler la définition classique de l'isomorphisme de graphes (voir la partie 3.2.1), qui s'intéresse au respect des arêtes existant entre les sommets.

Définition 5.7 (Isomorphisme de graphes)
Deux graphes $G_1 = (V_1, E_1)$ *et* $G_2 = (V_2, E_2)$ *sont* isomorphes *s'il existe une bijection* ϕ *de* V_1 *dans* V_2 *telle que*

$$\{i, j\} \in E_1 \Leftrightarrow \{\phi(i), \phi(j)\} \in E_2. \tag{5.1}$$

Ceci signifie que les deux graphes de la figure 5.xiii sont isomorphes, puisque l'appariement dénoté par les couleurs respecte toutes les arêtes. Mais nous l'avons dit : pour nos domaines d'application, il faut que les faces soient respectées, ce qui n'est pas le cas ici. En effet, le graphe de gauche a trois faces, de longueurs respectives 3, 6 et 5, tandis que celui de droite est composé de faces de longueurs 3, 4 et 7. Cet isomorphisme n'est donc pas assez contraint.

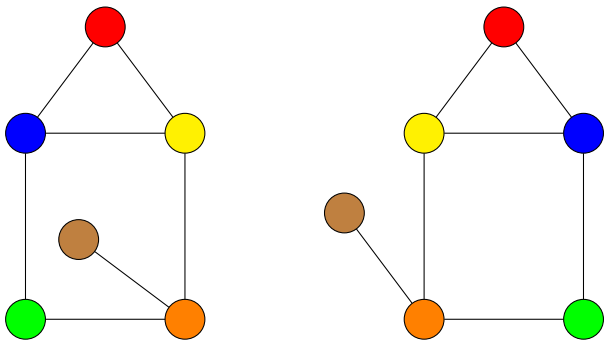


FIG. 5.xiii – Deux graphes isomorphes au sens classique du terme.

5.3.2 Isomorphisme sphérique

La nécessité de préservation des faces nous amène à définir un isomorphisme que nous qualifierons de *sphérique*. Nous verrons la raison de cette nomenclature par la suite.

Définition 5.8 (Isomorphisme sphérique)
Deux SDGP bien fondés $G = (V, F, e)$ *et* $G' = (V', F', e')$ *sont* sphères-isomorphes, *ce que nous noterons* $G \equiv_s G'$, *s'il existe :*

1. *une bijection* $\chi : V \to V'$ *sur les sommets ;*
2. *un appariement* $\xi : F \to F'$ *sur les faces tel que :*
   (a) ξ *est une bijection ;*

(b) $\forall f \in F$, $f' \in F'$, *si* $\xi(f) = f'$ *et que* $\mathbf{f} = [v_0 v_1 \dots v_n]$, *alors* $\mathbf{f}' = [\chi(v_0)\chi(v_1)\dots\chi(v_n)]$.

Les deux graphes de la figure 5.xiii ne sont pas sphères-isomorphes. En effet, le graphe de droite a une face de degré 4, alors que celui de gauche n'en a aucune. Par contre, les graphes de la figure 5.xiv le sont. Effectivement, on remarque que le parcours des faces (effectué en laissant ce qui ne leur appartient pas sur la droite) est bien toujours le même, quelque soit le plongement considéré (par exemple, la face de longueur 3 a pour mot circulaire [*rouge bleu jaune*] dans tous les cas).

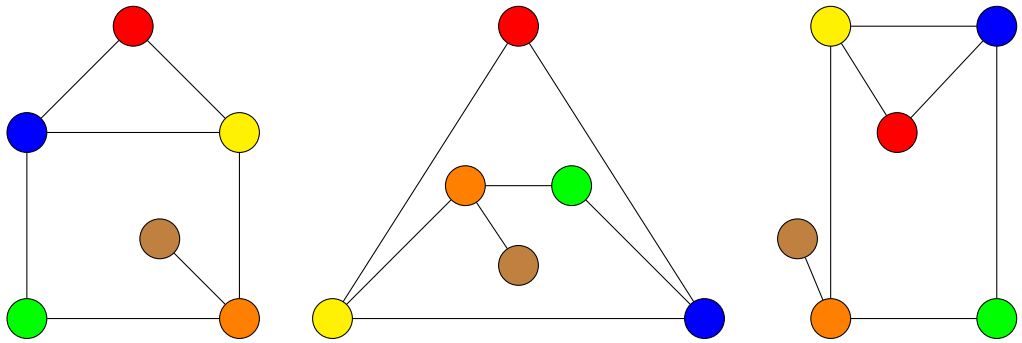


FIG. 5.xiv – Trois graphes sphères-isomorphes.

Néanmoins, l'isomorphisme sphérique n'est pas encore suffisamment contraint pour aborder nos problèmes de recherche de motifs. En fait, comme notre définition n'impose pas de contrainte de préservation de la face extérieure, toutes les faces jouent le même rôle. Donc tout se passe comme si les graphes étaient dessinés sur des sphères, plutôt que sur un plan. De fait, si l'on dessine l'un d'eux sur une sphère, on obtient la figure 5.xv. Or, il suffit de changer de point de vue en tournant autour de la sphère pour obtenir les représentations des deux autres graphes. C'est pour cette raison que l'on emploie le terme d'isomorphisme sphérique.

5.3.3 Isomorphisme plan

Finalement, la notion d'isomorphisme la plus intéressante est celle qui préserve à la fois les faces et la face externe. Nous le définissons de la manière suivante :

Définition 5.9 (Isomorphisme plan)
Deux SDGP bien fondés $G = (V, F, e)$ *et* $G' = (V', F', e')$ *sont* plans-isomorphes, *ce que nous noterons* $G \equiv_p G'$, *s'il existe :*

1. *une bijection* $\chi : V \to V'$ *sur les sommets ;*
2. *un appariement* $\xi : F \to F'$ *sur les faces tel que :*

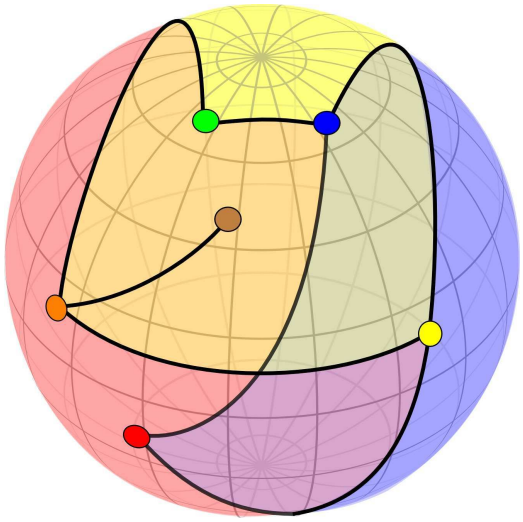


FIG. 5.xv – Graphe à trois faces (rouge, jaune et bleue) dessiné sur une sphère.

(a) ξ *est une bijection ;*

(b) $\forall f \in F,\ f' \in F',$ *si* $\xi(f) = f'$ *et que* $\mathbf{f} = [v_0 v_1 \dots v_n]$, *alors* $\mathbf{f'} = [\chi(v_0)\chi(v_1)\dots\chi(v_n)]$ *;*

(c) ξ *préserve la face externe :* $\xi(e) = e'$.

Les graphes de la figure 5.xiv sont sphères-isomorphes, mais pas plans-isomorphes, au contraire des graphes de la figure 5.xvi. Sur ces derniers, toutes les faces, y compris la face externe, sont préservées.

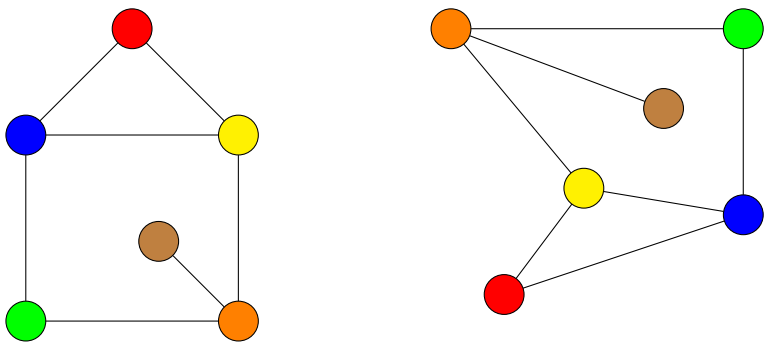


FIG. 5.xvi – Deux graphes plans-isomorphes.

5.3.4 Extension aux graphes plans à trous

Lorsque l'on considère des graphe plans à trous, on s'attend à ce que les isomorphismes préservent la propriété de visibilité des faces. Nous adaptons donc les définitions d'isomorphisme sphérique et d'isomorphisme plan de façon adéquate.

Définition 5.10 (Isomorphisme sphérique de graphes plans à trous)
Deux graphes plans à trous $G = (V, E, F, F_v, e)$ *et* $G' = (V', E', F', F_v', e')$ *sont* sphères-isomorphes*, ce que nous noterons* $G \equiv_s G'$*, si :*

1. *les deux SDGP* $G = (V, F, e)$ *et* $G' = (V', F', e')$ *sont sphères-isomorphes par les bijections* χ *et* ξ *;*
2. ξ *préserve les faces visibles :* $\xi(F_v) = F_v'$ *(et par conséquent,* $\xi(F \backslash F_v) = F' \backslash F_v'$*).*

Définition 5.11 (Isomorphisme plan de graphes plans à trous)
Deux graphes plans à trous $G = (V, E, F, F_v, e)$ *et* $G' = (V', E', F', F_v', e')$ *sont* plans-isomorphes*, ce que nous noterons* $G \equiv_p G'$ *si :*

1. *les deux SDGP* $G = (V, F, e)$ *et* $G' = (V', F', e')$ *sont plans-isomorphes par les bijections* χ *et* ξ *;*
2. ξ *préserve les faces visibles :* $\xi(F_v) = F_v'$ *(et par conséquent,* $\xi(F \backslash F_v) = F' \backslash F_v'$*).*

Clairement, deux graphes plans-isomorphes sont également sphères-isomorphes, mais le contraire n'est pas forcément vrai.

5.4 Équivalence de graphes plans à trous

Revenons sur les notions de visibilité et invisibilité des faces. Nous avons déjà expliqué le fait que les faces visibles correspondent au(x) motif(s) recherché(s). Ceci signifie qu'il est primordial que les faces visibles, ainsi que leur agencement, soient identiques pour décréter que deux motifs sont les mêmes. Par contre, les faces invisibles, elles, ne sont pas soumises aux mêmes contraintes. Or, nous n'avons pas fait cette distinction dans les définitions d'isomorphismes de graphes plans, pour lesquels les faces invisibles doivent être isomorphes entre elles, tout comme les faces visibles. Pour y remédier, nous allons dans cette partie définir l'*équivalence* de graphes plans à trous, qui serait un isomorphisme de graphes plans ne tenant pas compte des faces invisibles.

5.4.1 Définition

Considérons les graphes plans de la figure 5.xvii. Ces deux graphes plans à trous ne sont pas plans-isomorphes. Plusieurs différences sont remarquables : des arêtes et des sommets (en vert) ont été ajoutés (ou supprimés) et le nombre de faces n'est pas le même. Pourtant, il faut signaler que toutes ces différences n'ont affecté que les faces invisibles. En effet, si l'on ne considère que les faces visibles (ensemble des faces grises, bleue et jaune), alors les graphes sont identiques : ils sont *équivalents*.

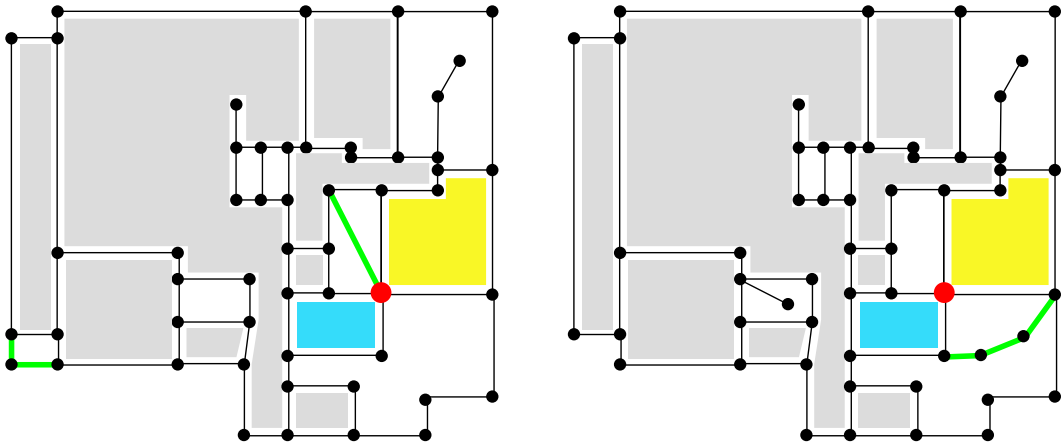


FIG. 5.xvii – Ces deux graphes plans à trous sont plans-isomorphes si l'on ne considère que les faces visibles : ils sont équivalents.

Définition 5.12 (Équivalence de graphes plans à trous)
Deux graphes plans à trous $G = (V, E, F, F_v, e)$ *et* $G' = (V', E', F', F_v', e')$ *sont* équivalents, *ce que nous noterons* $G \cong G'$, *s'il existe une paire* $\langle \phi, (\psi_f)_{f \in F_v} \rangle$ *telle que :*

1. $\phi : F_v \to F_v'$ *est une bijection sur les faces visibles ;*
2. $(\psi_f)_{f \in F_v}$ *est une famille d'applications telle que si* $\phi(f) = f'$ *pour deux faces* $f \in F_v, f' \in F_v'$, *alors :*
   (a) $\psi_f : V_f \to V'_{f'}$ *est une bijection des sommets de la frontière de* f *aux sommets de la frontière de* f', *et*
   (b) *si* $\mathbf{f} = [v_0 v_1 \dots v_n]$, *alors* $\mathbf{f}' = [v'_0 v'_1 \dots v'_m]$, *avec* $n = m$ *et* $\forall i \in \{0, 1, \dots, n\}$, $v'_i = \psi_f(v_i)$.
3. *Deux faces visibles de* G *ayant une arête commune auront une image ayant une arête commune correspondante dans* G'. *Ainsi, soit* $a = \{x, y\} \in E$ *une arête. Si* $faces(\{x, y\}) = \{f_1, f_2\}$, *avec* $f_1 \in F_v$, *alors*
   – *soit* $f_2 \in F_v$ *et* $\psi_{f_1}(x) = \psi_{f_2}(x)$ *et* $\psi_{f_1}(y) = \psi_{f_2}(y)$, *et donc* $faces(\{\psi_{f_1}(x), \psi_{f_1}(y)\}) = \{\phi(f_1), \phi(f_2)\}$,
   – *soit* $f_2 \notin F_v$ *et* $faces(\{\psi_{f_1}(x), \psi_{f_1}(y)\}) = \{\phi(f_1), g\}$ *avec* $g \notin F_v'$.
4. *la face externe est préservée :*
   – *si* $e \in F_v$, *alors* $\phi(e) = e'$ *et* $e' \in F_v'$;
   – *si* $e \in F \setminus F_v$, *alors* $e' \in F \setminus F_v'$.

Ainsi, les graphes de la figure 5.xviii sont équivalents : toutes les faces visibles sont respectées, ainsi que leurs arêtes communes. On notera cependant que la face extérieure du premier graphe ne peut pas être mise en correspondance avec celle du second graphe. Néanmoins, un robot qui se déplacerait à l'intérieur des faces visibles du premier graphe et du second n'a aucun moyen de les distinguer.

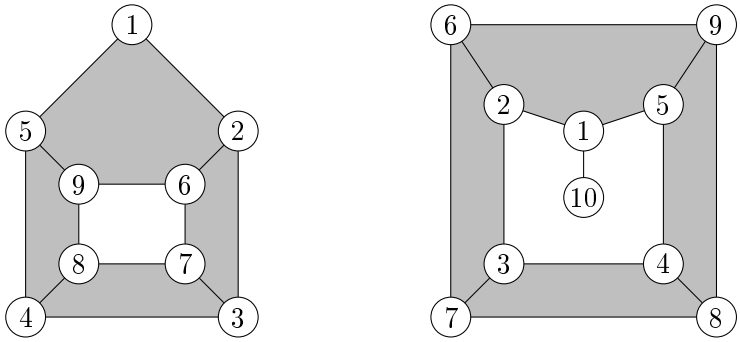


FIG. 5.xviii – Deux graphes plans à trous équivalents.

Les graphes de la figure 5.xix ne sont, eux, pas équivalents : toutes les arêtes communes des faces visibles ne sont pas respectées, ni même les faces externes qui, ici, doivent être prises en compte, puisqu'elles sont visibles.

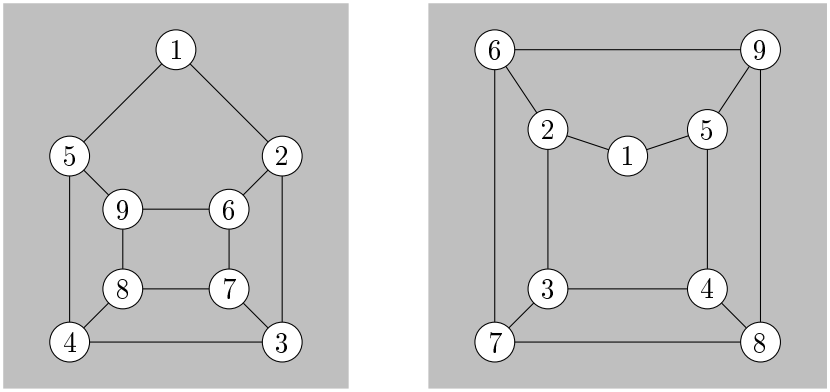


FIG. 5.xix – Deux graphes plans à trous non équivalents.

Quant aux graphes de la figure 5.xx, ils sont équivalents, puisqu'on peut définir une bijection sur les faces visibles dont on peut déduire, pour chaque face, une bijection sur les sommets, et qui satisfont toutes les contraintes de la définition.

Enfin, les graphes de la figure 5.xxi ne sont pas équivalents car un robot évoluant sur les faces visibles est en mesure d'observer la différence entre les deux.

On a le résultat suivant :

Théorème 5.1 *La relation $\cong$ est une relation d'équivalence.*

Preuve : La réflexivité et la transitivité sont évidentes. Concernant la symétrie, supposons que le graphe $G = (V, E, F, F_v, e)$ est équivalent au graphe $G' = (V', E', F', F_v', e')$

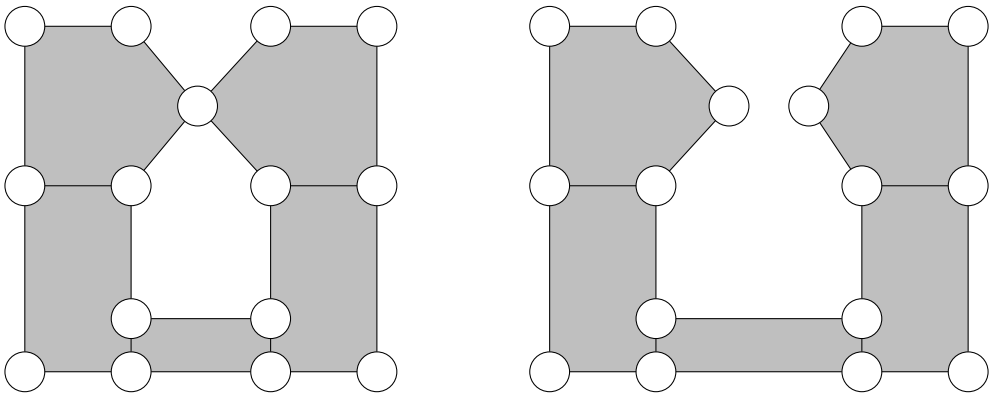


FIG. 5.xx – Deux graphes plans à trous équivalents.

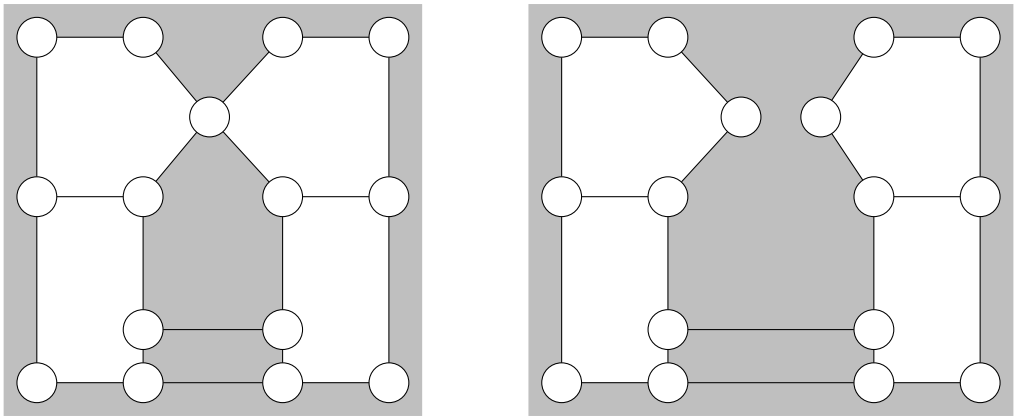


FIG. 5.xxi – Deux graphes plans à trous non équivalents.

par $\langle \phi, (\psi_f)_{f \in F_v} \rangle$. Nous définissons la paire $\langle \phi', (\psi_g)_{g \in {F_v}'} \rangle$ pour prouver que $G' \cong G$. Comme $\phi : F_v \to {F_v}'$ est une bijection sur les faces visibles, on pose $\phi' = \phi^{-1}$. Maintenant, pour toute face $g \in {F_v}'$, nous définissons ψ_g. La face g a un unique antécédent f par ϕ, et comme ψ_f est une bijection de sommets de $\mathbf{f}$ vers les sommets de $\mathbf{g}$, on pose $\psi'_g = \psi_f^{-1}$. Clairement, ψ'_g est une bijection des sommets de $\mathbf{g}$ vers les sommets de $\mathbf{f}$ qui préserve la frontière de g. Finalement, soit une arête $a' = \{x, y\} \in E'$ et soit $\{g, g'\} = faces(a')$ avec $g \in {F_v}'$. On a $\{\psi'_g(x), \psi'_g(y)\} = a \in E$ et $faces(a) = \{\phi'(g), f'\}$; de plus, $g' \in {F_v}'$ si et seulement si $f' \in F_v$. Finalement, on sait que si $f' \in F_v$ alors $\psi_f(\psi'_g(x)) = \psi_{f'}(\psi'_g(x))$, c'est-à-dire, $x = \psi_{f'}(\psi'_g(x))$, ainsi $\psi'_{g'}(x) = \psi'_g(x)$. Et, pour la même raison, $\psi'_g(y) = \psi'_{g'}(y)$. □

5.4.2 Passerelles, charnières et arêtes pendantes

Il existe manifestement des liens étroits entre isomorphisme et équivalence. En fait, ces notions seront les mêmes lorsque les faces invisibles auront été vidées de leur contenu, et les graphes *normalisés* (en un sens à préciser). Revenons à l'application de modélisation de l'environnement pour des robots. Le postulat est le suivant : deux graphes sont équivalents si un robot se déplaçant de pièce en pièce (donc de face visible en face visible) n'est pas capable de les distinguer. Autrement dit, si l'on vide le contenu des faces invisibles, alors deux graphes équivalents le resteront.

Nous étudions maintenant le contenu des faces invisibles, et pour cela nous introduisons deux entités particulières : les passerelles et les charnières.

Définition 5.13 (Passerelle)
Une passerelle *d'un graphe plan à trous* $G = (V, E, F, F_v, e)$ *est une arête* $a \in E$ *telle que* $|faces(a)| = 2$ *et* $faces(a) \cap F_v = \emptyset$.

Les graphes de la figure 5.xvii contiennent plusieurs passerelles, représentées en couleur verte. La figure 5.xxii met en avant leur rôle : elles délimitent deux faces invisibles (délimitation des faces f_1 et f_2). Pourquoi distinguer les passerelles des autres arêtes ? Clairement, un robot ne voit pas l'intérieur des faces invisibles, il ne sait donc pas si elles sont divisées ou non en plusieurs faces. L'ajout ou la suppression de passerelles ne modifie en rien sa perception de l'espace. Ce sont donc des arêtes dont l'(in)existence lui est indifférente.

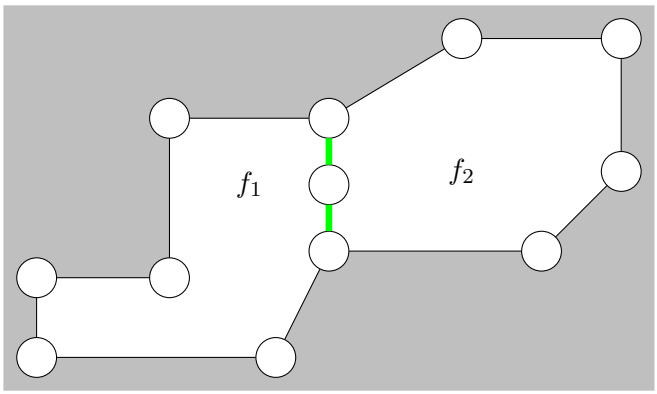


FIG. 5.xxii – Délimitation de deux faces invisibles par deux passerelles (arêtes vertes).

Le second objet, appelé charnière, concerne les sommets des graphes plans à trous.

Définition 5.14 (Charnière)
Un sommet $v \in V$ *d'un graphe plan à trous* $G = (V, E, F, F_v, e)$ *est une* charnière *si son voisinage* $\Gamma(v) = [y_1 y_2 \dots y_n]$ *est tel qu'il existe* $1 \leq i < j < k < l \leq n$ *tels que* $\overrightarrow{vy_i} \in F_v, \overrightarrow{vy_j} \in F \backslash F_v, \overrightarrow{vy_k} \in F_v, \overrightarrow{vy_l} \in F \backslash F_v$.

Ainsi, les graphes de la figure 5.xvii page 127 ont tous deux la même charnière, qui est le sommet de couleur rouge. Tout sommet devient charnière dès qu'il existe au moins deux faces invisibles non consécutives autour de lui (voir par exemple la figure 5.xxiii). Pourquoi distinguer les charnières des autres sommets ? Imaginons que le robot se trouve dans la face bleue (figure 5.xvii). Il connaît donc le sommet rouge, qui est pour lui un des angles de la pièce. En se déplaçant de face visible en face visible, il peut atteindre la face jaune. Or, il n'a aucun moyen de savoir que le sommet rouge est le même que celui de la face bleue (les faces invisibles empêchent une bonne visibilité) : pour lui, tout se passe comme s'il s'agissait d'un nouveau sommet.

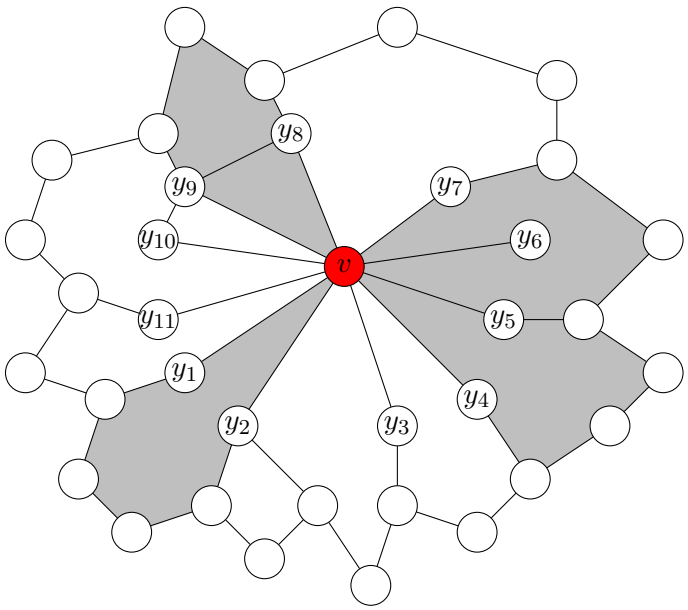


FIG. 5.xxiii – Le sommet rouge est une charnière.

Terminons par la définition d'une arête pendante :

Définition 5.15 (Arête pendante)
Un sommet $v \in V$ est dit pendant *s'il est incident à une seule arête : $\delta(v) = 1$. Une arête $a \in E$ est* pendante *si elle est incidente à un sommet pendant.*

5.4.3 Normalisation des graphes à trous

Les passerelles et les charnières nous permettent de mettre en relation l'isomorphisme et l'équivalence des graphes plans à trous. En effet, éliminer, par un processus que nous allons étudier, ces deux objets, maintient l'équivalence des graphes considérés. Mais, si ce processus est de plus couplé à la suppression des arêtes pendantes des faces invisibles, alors il nous permet d'obtenir une forme irréductible unique pour une classe de graphes

équivalents. En définitive, tester si deux graphes plans à trous sont équivalents se ramène à tester si leurs formes irréductibles vérifient un des isomorphismes que nous avons définis. Nous commençons dans cette partie par expliciter le processus d'élimination des charnières, passerelles, et arêtes pendantes des faces invisibles.

Soit la définition d'un graphe irréductible suivante :

Définition 5.16 (Graphe irréductible)
Un graphe plan à trous $G = (V, E, F, F_v, e)$ est irréductible *s'il ne contient ni charnière, ni passerelle, ni arête pendante dans une face invisible.*

La forme irréductible d'un graphe plan à trous G sera notée $\widehat{G}$. Nous allons proposer un algorithme qui, étant donné un graphe plan à trous, retourne le graphe irréductible correspondant. Ce graphe est une forme normale unique, et lui est équivalent. Ce processus est appelé normalisation, et consiste dans un premier temps à supprimer les charnières, dans un deuxième temps les passerelles, et finalement à éliminer les arêtes pendantes des faces invisibles.

Nous avons fait une distinction entre les arêtes pendantes et les passerelles. Rappelons que ces dernières sont des arêtes séparant deux faces invisibles. Il reste néanmoins un cas que nous n'avons pas évoqué, celui des *pseudo-passerelles*, c'est-à-dire des arêtes qui ne sont adjacentes qu'à une face invisible sans être pendantes pour autant (comme par exemple l'arête jaune de la figure 5.xxiv).

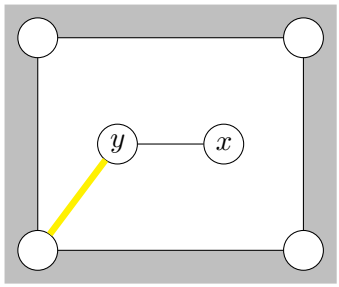


FIG. 5.xxiv – Une pseudo-passerelle (en jaune).

En fait, grâce au résultat suivant, ces pseudo-passerelles ne seront pas gênantes dans le processus de normalisation, quand nous éviderons le contenu des faces invisibles. De façon plus précise, toute pseudo-passerelle est appelée à devenir une arête pendante en cours de normalisation, et sera donc éliminée par la procédure :

Proposition 5.1 *Soit G un graphe plan à trous, f une face invisible et $a = \{x, y\}$ une arête telle que $faces(a) = \{f\}$. Alors soit G admet une arête pendante dans une face invisible, soit G admet une passerelle.*

Preuve : S'il existe $u \in V^*$ tel que $\mathbf{f} = [xyuy]$ avec $u \in V^*$, alors $\{x, y\}$ est une arête pendante (c'est le cas représenté sur la figure 5.xxiv). Sinon, il existe $u, u' \in V^*$ tels que $\mathbf{f} = [xyuyxu']$ (comme sur la figure 5.xxv). Or, il existe au moins une zone du plan délimitée par $[yu]$ ou $[xu']$ qui ne contienne pas de face visible, sinon ces dernières ne seraient pas contiguës. Supposons sans perte de généralité qu'il s'agisse de $[yu]$, et que $u = v_1 v_2 \dots v_n$. Alors il y a deux cas. (1) Soit toutes les lettres (sommets) de u sont distinctes, ce qui est le cas de l'illustration du haut de la figure 5.xxv. Alors $[yu]$ désigne un cycle élémentaire du graphe. Aussi, la face $g = \overrightarrow{v_1 y}$ est une face invisible distincte de $f = \overrightarrow{y v_1}$. Donc l'arête $\{y, v_1\}$ est une passerelle de G. (2) Sinon (illustration du bas de la figure 5.xxv), on peut décomposer u sous la forme $u_1 v v' u_2 v u_3$ avec $v, v' \in V$ et $u_1, u_2, u_3 \in V^*$, de sorte que toutes les lettres de u_2 soient distinctes. Alors clairement, soit u_2 est vide, et dans ce cas $\{v, v'\}$ est une arête pendante dans la face invisible f, soit u_2 n'est pas vide, et dans ce cas $\{v, v'\}$ est une passerelle, comme dans le cas (1). □

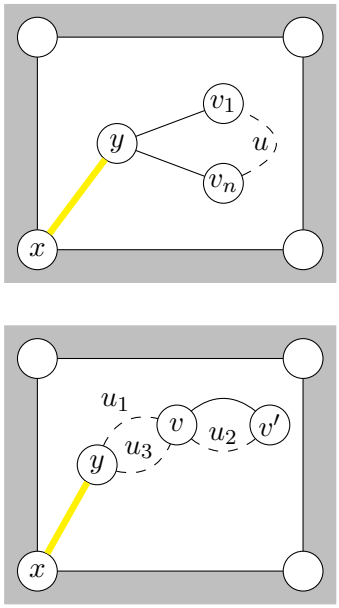


FIG. 5.xxv – Illustration du second cas de la preuve. En haut, *(1)*, en bas, *(2)*.

Nous nous intéressons maintenant plus en détails aux différentes étapes de la normalisation, réalisées par l'algorithme 1.

Élimination des charnières

Lorsque l'on élimine une charnière, on *ouvre* cette dernière, c'est-à-dire que l'on désunit les faces qui étaient liées par elle. Ce processus engendre plusieurs modifications

Algorithme 1: NORMALISERGRAPHE(G)

Données : un graphe plan à trous $G = (V, E, F, F_v, e)$
Résultat : le graphe irréductible correspondant

```
1  tant que il existe une charnière dans G faire
2      soit v la charnière;
3      ELIMINERCHARNIERE(G, v) ;
4  tant que il existe une passerelle dans G faire
5      soit p = {x, y} la passerelle;
6      ELIMINERPASSERELLE(G, p) ;
7  tant que il existe une arête pendante dans une face invisible de G faire
8      soit a = {x, y} l'arête pendante;
9      ELIMINERARETEPENDANTE(G, a) ;
10 retourner le graphe irréductible Ĝ
```

du graphe : à chaque charnière supprimée, les ensembles des sommets, des arêtes et des faces changent. C'est le principe de l'algorithme 2.

Ainsi, considérons le graphe plan à trous de la figure 5.xxiii. La charnière (en rouge) est v, et $\Gamma(v) = [y_1 y_2 \dots y_{11}]$ (on a bien $\overrightarrow{vy_1} \in F_v$ et $\overrightarrow{vy_{11}} \in F \setminus F_v$). La figure 5.xxvi présente le résultat de l'élimination de cette charnière. Le sous-mot le plus long sélectionné (lignes 2 à 5 de l'algorithme) est $[y_1 y_2 y_4 y_7 y_8 y_9]$. La nouvelle face créée est $\mathbf{h} = [y_1 v_1 y_2 v_2 y_4 v_4 y_7 v_7 y_8 v_8 y_9 v_9]$.

On remarque que la suppression d'une charnière engendre la création de nombreuses passerelles. De plus, ceci peut paraître paradoxal, mais notons que l'élimination des charnières crée de nouveaux sommets. Cela signifie que la forme irréductible d'un graphe peut avoir une taille supérieure au graphe non normalisé.

Lemme 5.1 *Soit $G = (V, E, F, F_v, e)$ un graphe plan à trous et $v \in V$ une charnière. Alors le graphe G' obtenu après l'appel à l'algorithme 2 pour éliminer v est équivalent à G.*

Preuve : Chaque face visible $f = \overrightarrow{vy_j}$ avec $j \in \{1, 2 \dots n\}$ est transformée en une face f' dont la frontière $\mathbf{f'}$ est déduite de $\mathbf{f}$ en remplaçant toutes les occurrences de v par v_p. Donc pour de telles faces, on fixe $\phi(f) = f'$ et $\psi_f(v) = v_p$ et $\psi_f(y) = y$ pour tout $y \neq v$. En ce qui concerne les autres faces visibles, on fixe $\phi(f) = f$ et $\psi_f(y) = y$ pour tout $y \in V_f$. Clairement, la condition *1.* de la définition 5.12 est vérifiée parce que l'algorithme 2 ne modifie pas le statut (visibilité ou invisibilité) des faces existantes. La condition *2.* est vérifiée car tous les v_p sont de nouveaux sommets, donc tous les ψ_f sont bien des bijections qui préservent les frontières. Pour la même raison, la condition *4.* est vérifiée.

Considérons maintenant une arête a et soit $\{f, g\} = faces(a)$. La condition *3.* est immédiate dans tous les cas, excepté celui où $f = \overrightarrow{vy_j}$ et $g = \overrightarrow{vy_{j+1}}$ et $a = \{v, y_{j+1}\}$ pour

Algorithme 2: ELIMINERCHARNIERE(G, v)

Données : un graphe plan à trous $G = (V, E, F, F_v, e)$, une charnière $v \in V$
Résultat : le graphe modifié G

soit $[y_1 y_2 \dots y_n] = \Gamma(v)$ le voisinage du sommet v, tel que $\overrightarrow{vy_1} \in F_v$ et $\overrightarrow{vy_n} \in (F \setminus F_v)$;
sélectionner le sous-mot le plus long $[y_{i_1} y_{i_2} \dots y_{i_k}]$ de $\Gamma(x)$ tel que
    $1 = i_1 < i_2 < i_3 < \dots i_k \leq n$, et $\forall s \in \{1, 2, \dots, k-1\}$
    soit $(\overrightarrow{vy_{i_s}} \in F_v$ et $\overrightarrow{vy_{i_s+1}} \in (F \setminus F_v))$
    soit $(\overrightarrow{vy_{i_s}} \in (F \setminus F_v)$ et $\overrightarrow{vy_{i_s+1}} \in F_v)$;
`// chaque arête` $\{v, y_{i_s}\}$ `sépare une face visible et une face invisible`
$V \leftarrow V \setminus \{v\} \cup \{v_{i_1}, v_{i_2}, \dots v_{i_k}\}$;
pour chaque face $f = \overrightarrow{vy_j}$ avec $j \in 1, 2 \dots n$ **faire**
    soit p le plus grand indice dans $\{i_1, i_2, \dots i_k\}$ tel que $p \leq j$;
    remplacer toutes les occurrences de v par v_p dans $\mathcal{D}(f)$;
$\mathbf{h} \leftarrow [y_{i_1} v_{i_1} y_{i_2} v_{i_2} \dots y_{i_k} v_{i_k}]$;
$F \leftarrow F \cup \{h\}$;
`//` h `est une face invisible`
mettre à jour E en parcourant F ;
retourner le graphe modifié G

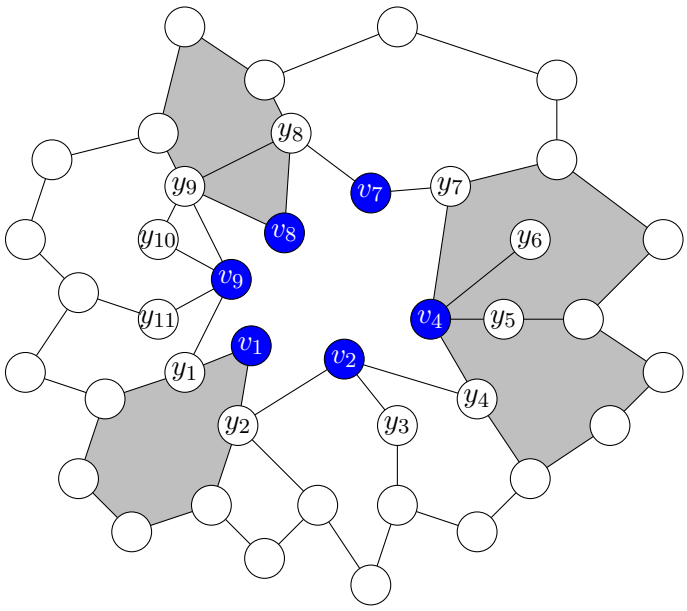


FIG. 5.xxvi – Graphe résultant de l'élimination de la charnière de la figure 5.xxiii page 131. Les nouveaux sommets sont en bleu.

un certain j. On suppose, sans perte de généralité, que f est visible, et que le sommet v de $\mathbf{f}$ est remplacé par un certain v_p dans la frontière de $\mathbf{f}'$. Dans ce cas, l'arête $\{v, y_{j+1}\}$ est transformée en une arête $\{v_p, y_{j+1}\}$. Alors il y a deux cas. Soit g est elle aussi visible, et alors p est le plus grand indice dans $\{i_1, i_2, \dots i_k\}$ tel que $p \leq j+1$; alors v va aussi être remplacé par v_p dans $\mathbf{g}$. Donc on a $\psi_f(v) = v_p = \psi_g(v)$ et $\psi_f(y_{j+1}) = y_{j+1} = \psi_g(y_{j+1})$. Dans le second cas, g n'est pas visible ; alors $\{v_p, y_{j+1}\}$ est une arête de la nouvelle face invisible h qui est ajoutée au graphe. En d'autres termes, $faces(\{v_p, y_{j+1}\}) = \{\phi(f), h\}$ et h est bien invisible. □

Élimination des passerelles

Rappelons qu'une passerelle est une arête qui sépare deux faces invisibles distinctes. La supprimer avec l'algorithme 3 revient à fusionner ces deux faces pour n'en obtenir qu'une seule dont on doit calculer la nouvelle frontière.

Algorithme 3: ELIMINERPASSERELLE(G, p)

Données : un graphe plan à trous $G = (V, E, F, F_v, e)$, une passerelle $p = \{x, y\} \in E$
Résultat : G modifié

1 soient $\{f, f'\} = faces(\{x, y\})$ avec $\mathbf{f} = [yxu]$ et $\mathbf{f}' = [vxy]$ pour $u, v \in V^*$;
2 $\mathbf{f} \leftarrow [yvxu]$;
3 $F \leftarrow F \setminus \{f'\}$;
4 **si** $e = f'$ **alors**
5 $\quad e \leftarrow f$;
6 $E \leftarrow E \setminus \{x, y\}$;
7 **retourner** le graphe modifié G

La figure 5.xxvii montre le résultat de la suppression d'une des deux passerelles de la figure 5.xxii. La face f_2 n'existe plus, et a fusionné avec f_1. Avant ce processus, ce graphe comptait deux passerelles. Après la suppression de l'une d'entre elles, il n'en contient plus aucune : la seconde passerelle est devenue une arête pendante.

Lemme 5.2 *Soit $G = (V, E, F, F_v, e)$ un graphe plan à trous et $p = \{x, y\} \in E$ une passerelle. Alors le graphe G' obtenu après l'appel à l'algorithme 3 pour éliminer p est équivalent à G.*

Preuve : L'algorithme 3 n'a aucune influence sur les faces visibles. En effet, aucun sommet n'est éliminé. Concernant les arêtes, seule celle correspondant à la passerelle est supprimée. Or, par définition, une passerelle est adjacente à deux faces invisibles. Sa suppression engendre donc l'élimination d'une face invisible et la modification d'une autre. Les faces visibles ne sont pas concernées. De même, e ne change que si elle correspond à la face invisible modifiée, mais reste dans tous les cas invisible. On fixe donc ϕ (bijection sur les faces visibles) comme étant l'identité, et on a bien G' équivalent à G. □

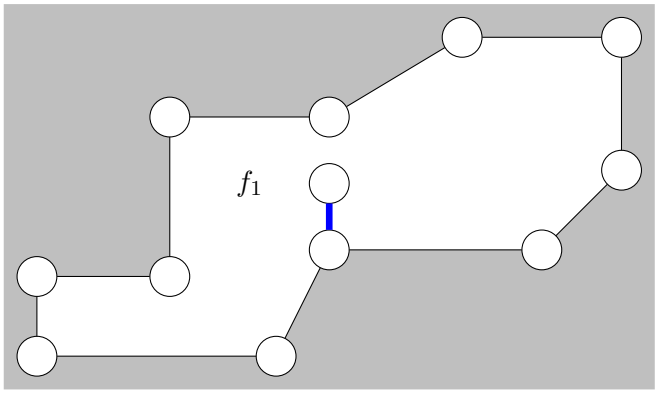


FIG. 5.xxvii – Graphe résultant de l'élimination d'une des passerelles de la figure 5.xxii. Il n'y a plus aucune passerelle, mais une arête pendante a été créée (en bleu). Les deux faces ont fusionné.

On notera que l'élimination d'une passerelle n'introduit pas de nouvelle charnière. En effet, la seule façon de créer une charnière est de désolidariser deux faces visibles contiguës, ce que ne fait en aucun cas le processus d'élimination des passerelles.

Élimination des arêtes pendantes des faces invisibles

Il peut exister des arêtes pendantes à l'intérieur des faces invisibles. De plus, nous l'avons vu, la suppression des passerelles peut aussi en créer. Or, l'(in)existence d'arêtes pendantes à l'intérieur des faces invisibles n'a, tout comme les passerelles, aucun impact sur la perception de l'environnement du robot. L'algorithme 4, dernière phase de la normalisation, vise à les supprimer pour obtenir la forme normalisée.

Algorithme 4: ELIMINERARETEPENDANTE(G, a)

Données : un graphe plan à trous $G = (V, E, F, F_v, e)$, une arête pendante d'une face invisible $a = \{x, y\} \in E$ avec $|\Gamma(x)| = 1$
Résultat : G sans l'arête pendante a

1 soit $f = faces(\{x, y\})$;
2 on suppose que $\mathbf{f} = [uyxy]$ avec $u \in V^*$;
3 $\mathbf{f} \leftarrow [uy]$;
4 $V \leftarrow V \setminus \{x\}$;
5 $E \leftarrow E \setminus \{x, y\}$;
6 **retourner** le graphe modifié G

Lemme 5.3 *Soit $G = (V, E, F, F_v, e)$ un graphe plan à trous et $a = \{x, y\} \in E$ une*

arête pendante d'un face invisible. Alors le graphe G' obtenu après l'appel à l'algorithme 4 pour éliminer a est équivalent à G.

Preuve : L'algorithme 4 n'a aucune influence sur les faces visibles. Donc de nouveau, le résultat est immédiat. □

Notons que l'élimination des arêtes pendantes ne crée pas de charnière (pour la même raison que l'élimination des passerelles), et ne crée pas non plus de passerelle, puisque la seule façon d'introduire une passerelle serait de diviser une face invisible en deux, ce qui n'est pas le cas de ce processus.

Résultat final

La figure 5.xxviii représente le résultat de la normalisation des graphes de la figure 5.xvii : tous deux mènent au même graphe irréductible, qui leur est équivalent.

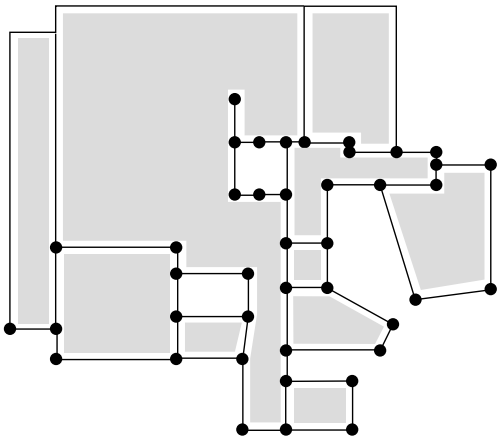


FIG. 5.xxviii – Le graphe normalisé correspondant aux graphes de la figure 5.xvii.

Théorème 5.2 *Soit $G = (V, E, F, F_v, e)$ un graphe plan à trous. L'algorithme* NORMALISERGRAPHE(G) *permet d'obtenir un graphe irréductible équivalent à G en temps polynomial.*

Preuve : Chaque étape de la normalisation préserve l'équivalence. Par ailleurs, l'élimination d'une charnière introduit de nombreuses passerelles mais l'élimination d'une passerelle n'introduit pas de nouvelle charnière. De même, l'élimination des passerelles introduit des arêtes pendantes, mais l'élimination des arêtes pendantes n'introduit ni passerelle ni charnière. Enfin, chaque élimination d'une arête pendante diminue le nombre d'arêtes de 1, donc l'algorithme termine, en temps polynomial. □

5.4.4 Normalisation et connexité

Nous l'avons vu : la normalisation, et en particulier l'élimination d'une charnière, détruit certaines connexions du graphe d'origine. Cependant, le graphe normalisé reste connexe : la contrainte de contiguïté des faces visibles dans les graphes plans à trous est préservée à chaque étape du processus de normalisation. Au contraire, les faces visibles du graphe de la figure 5.xxix sont connexes mais non contiguës : dans ce cas, la normalisation le rendrait non connexe.

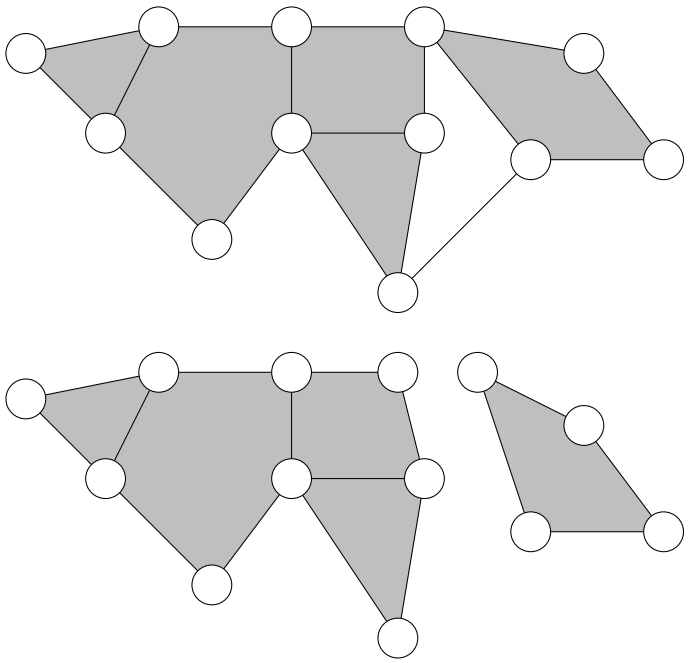


FIG. 5.xxix – En haut : un graphe dont les faces visibles ne sont pas contiguës, avant normalisation. En-dessous : le graphe irréductible correspondant, qui n'est plus connexe.

5.4.5 Relation entre équivalence et isomorphisme

Comme nous l'avons déjà évoqué, deux graphes plans équivalents seront isomorphes lorsqu'ils auront été normalisés. Néanmoins, la situation est un peu plus complexe, car il faut tenir compte du statut des faces externes.

Cas des faces externes invisibles

On a le résultat suivant :

Théorème 5.3 *Soient* $G_1 = (V_1, E_1, F_1, F_{v1}, e_1)$ *et* $G_2 = (V_2, E_2, F_2, F_{v2}, e_2)$ *deux graphes plans à trous. Si leurs faces externes sont invisibles, alors* G_1 *et* G_2 *sont équiva-*

lents si et seulement si leurs formes irréductibles sont sphères-isomorphes, c'est-à-dire

$$G_1 \cong G_2 \Leftrightarrow \widehat{G_1} \equiv_s \widehat{G_2}.$$

Ce résultat peut surprendre le lecteur : il s'agit bien d'un isomorphisme *sphérique*, et non pas d'un isomorphisme *plan*. En effet, considérons de nouveau la situation de la figure 5.xviii page 128 : les deux graphes sont bien équivalents, leurs formes irréductibles sont bien sphères-isomorphes mais leurs faces externes ne sont pas en bijection ; elles ne sont donc pas planes-isomorphes.

Commençons par le sens le plus simple :

Lemme 5.4 *Soient $G_1 = (V_1, E_1, F_1, F_{v1}, e_1)$ et $G_2 = (V_2, E_2, F_2, F_{v2}, e_2)$ deux graphes plans à trous dont les faces externes sont invisibles. Si $\widehat{G_1}$ et $\widehat{G_2}$ sont sphères-isomorphes, alors G_1 et G_2 sont équivalents.*

Preuve : Si $\widehat{G_1} \equiv_s \widehat{G_2}$, et que les faces externes sont invisibles, alors $\widehat{G_1} \cong \widehat{G_2}$. De plus, d'après le théorème 5.2, $\widehat{G_1} \cong G_1$ et $\widehat{G_2} \cong G_2$. Donc, $G_1 \cong \widehat{G_1} \cong \widehat{G_2} \cong G_2$. Par transitivité (théorème 5.1), on obtient : $G_1 \cong G_2$. □

Concentrons-nous maintenant sur la réciproque :

Lemme 5.5 *Soient $G_1 = (V_1, E_1, F_1, F_{v1}, e_1)$ et $G_2 = (V_2, E_2, F_2, F_{v2}, e_2)$ deux graphes plans à trous dont les faces externes sont invisibles. Si G_1 et G_2 sont équivalents alors $\widehat{G_1}$ et $\widehat{G_2}$ sont sphères-isomorphes.*

Preuve : Pour démontrer ce lemme, nous avons besoin de plusieurs notations. Nous disposons des graphes $G_1 = (V_1, E_1, F_1, F_{v1}, e_1)$ et $G_2 = (V_2, E_2, F_2, F_{v2}, e_2)$. Nous noterons $\widehat{G_1} = (\widehat{V_1}, \widehat{E_1}, \widehat{F_1}, \widehat{F_{v1}}, \widehat{e_1})$ et $\widehat{G_2} = (\widehat{V_2}, \widehat{E_2}, \widehat{F_2}, \widehat{F_{v2}}, \widehat{e_2})$ les formes irréductibles correspondantes. De plus, comme $G_1 \cong G_2$, soient $\phi : F_{v1} \to F_{v2}$ et $(\psi_g)_{g \in F_{v1}}$ les bijections issues de la définition 5.12. Par ailleurs, on sait que $\widehat{G_1} \cong G_1$ d'après le théorème 5.2 ; donc il existe des bijections $\phi'_1 : \widehat{F_{v1}} \to F_{v1}$ et $(\psi'_{1f})_{f \in \widehat{F_{v1}}}$. Enfin, comme $G_2 \cong \widehat{G_2}$, il existe des bijections $\phi : F_{v2} \to \widehat{F_{v2}}$ et $(\psi_{2h})_{h \in F_{v2}}$.

Maintenant, on veut prouver que $\widehat{G_1} \equiv_s \widehat{G_2}$, donc il nous faut définir $\xi : \widehat{F_1} \to \widehat{F_2}$ et $\chi : \widehat{V_1} \to \widehat{V_2}$ qui satisfont les conditions de la définition 5.8. La situation est représentée dans la figure 5.xxx.

Concernant la bijection ξ sur les faces, il y a deux cas. Logiquement, pour toute face visible f de $\widehat{G_1}$, on pose $\xi(f) = (\phi_2 \circ \phi \circ \phi'_1)(f)$. Comme ϕ'_1, ϕ et ϕ_2 sont des bijections, ξ est une bijection des faces visibles de $\widehat{G_1}$ vers les faces visibles de $\widehat{G_2}$. Malheureusement, on ne peut pas définir ξ sur les faces invisibles de la même façon, puisque ϕ'_1, ϕ et ϕ_2 ne sont pas définies pour ces faces-là. La seule alternative est donc de passer par les frontières des faces invisibles, donc d'utiliser la bijection χ sur les sommets.

Aussi, nous nous attachons maintenant à la définition de $\chi : \widehat{V_1} \to \widehat{V_2}$. Pour cela, et compte tenu des données du problème, il nous faut utiliser certaines bijections parmi $(\psi'_{1f})_{f \in \widehat{F_{v1}}}$, $(\psi_g)_{g \in F_{v1}}$ et $(\psi_{2h})_{h \in F_{v2}}$. Rappelons qu'elles sont attachées à des faces visibles uniquement.

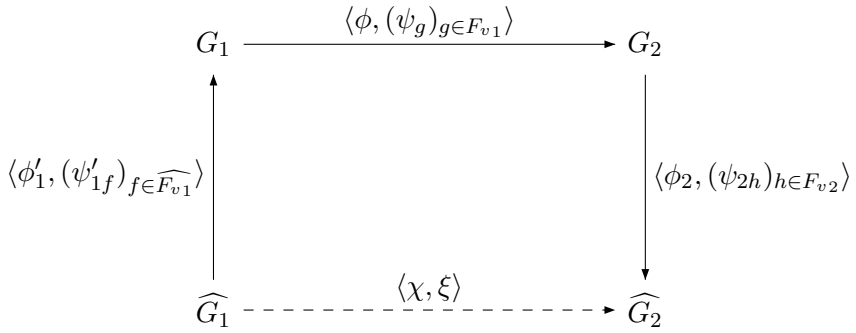


FIG. 5.xxx – Notations pour la preuve du lemme.

Soit v un sommet. Comme $\widehat{G_1}$ est normalisé, v participe à la frontière de (1) au moins une face visible, et (2) au plus une face invisible. En effet, le point (1) est vérifié car $\widehat{G_1}$ ne contient ni passerelle ni arête pendante dans des faces invisibles, et le point (2) l'est aussi, car $\widehat{G_1}$ ne contient pas de charnière. Soit donc f une face visible adjacente à v. On pose $g = \phi'_1(f)$ et $h = \phi(g)$. On définit maintenant $\chi(v) = (\psi_{2h} \circ \psi_g \circ \psi'_{1f})(v)$.

Montrons que cette définition est indépendante de la face f choisie. Supposons dans un premier temps que v participe à la frontière de deux faces visibles f et f' et que f et f' sont adjacentes : il existe une arête $\{v, y\}$ séparant f et f'. Par la condition *3.* de la définition 5.12, on a $\psi'_{1f}(v) = \psi'_{1f'}(v)$, et donc $\psi_{\phi'_1(f)}(\psi'_{1f}(v)) = \psi_{\phi'_1(f')}(\psi'_{1f'}(v))$, et donc $\chi(v)$ correspond bien à un unique sommet, que l'on choisisse f ou f' comme face visible initiale. Plus généralement, si f et f' ne sont pas adjacentes, alors il existe une séquence de faces visibles $f = f_1, f_2, \ldots, f_n = f'$ adjacentes à v et séparées deux à deux par des arêtes communes. Donc, par récurrence, l'image de $\chi(v)$ est là encore unique, quelle que soit la face visible choisie.

À ce stade, nous avons montré que χ était une fonction : tout sommet a bien une image unique. Pour montrer que c'est une bijection, il suffit de rappeler que $\cong$ est une relation d'équivalence. Aussi, toutes les flèches du diagramme de la figure 5.xxx pourraient être inversées en utilisant les bijections réciproques sur les faces visibles et les sommets. On obtiendrait donc de la même façon une définition de $\chi^{-1} : \widehat{V_2} \to \widehat{V_1}$ qui satisferait les mêmes propriétés que χ. Ainsi, pour tout sommet v' de $\widehat{G_2}$, il existe une unique image de v' par χ^{-1}. Autrement dit, tout sommet v' de $\widehat{G_2}$ a bien un unique antécédent par χ. Enfin, si f, f' sont des faces visibles et $\xi(f) = f'$, alors χ préserve bien les frontières de f : pour chaque sommet v de la frontière de f, on peut utiliser f comme face visible de base pour calculer $\chi(v)$; aussi, la condition *2.(b)* de la définition 5.12 nous permet de conclure.

Il nous reste à définir $\xi(f)$ pour les faces invisibles de $\widehat{G_1}$. Supposons que $\mathbf{f} = [v_0 v_1 \ldots v_n]$. Nous prétendons que $[\chi(v_0)\chi(v_1)\ldots\chi(v_n)]$ est la frontière d'une face invisible de $\widehat{G_2}$. En effet, considérons les arêtes $a_0 = \{v_0, v_1\}$ et $a_1 = \{v_1, v_2\}$. a_0 sépare la face invisible f d'une face visible g_0 car $\widehat{G_1}$ est normalisé. Par la condition *3.* de

la définition 5.12, $\{\chi(v_0), \chi(v_1)\}$ est une arête de $\widehat{G_2}$ qui sépare une face invisible h_0 de la face visible $\xi(g_0)$. De même, a_1 sépare la face invisible f d'une face visible g_1, et $\{\chi(v_1), \chi(v_2)\}$ est une arête qui sépare une face invisible h_1 de la face visible $\xi(g_1)$. Or, il existe au plus une face invisible adjacente au sommet $\chi(v_1)$ dans $\widehat{G_2}$. Donc $h_0 = h_1$. En refaisant le même raisonnement avec toutes les arêtes de la frontière de f (qui ne contient pas d'arête pendante), on en déduit que $[\chi(v_0)\chi(v_1)\dots\chi(v_n)]$ est bien la frontière d'une unique face invisible dans $\widehat{G_2}$, qu'on peut donc noter $\xi(f)$.

Avec cette définition, ξ est bien une bijection sur l'ensemble des faces invisibles (en plus d'être une bijection sur les faces visibles), et χ préserve leurs frontières, par construction.

□

Cas des faces externes visibles

Lorsque les faces externes sont visibles, par contre, on ne peut plus « retourner » les graphes (comme dans la figure 5.xviii page 128) en utilisant une face interne invisible du premier graphe comme face externe du second. Aussi, dans ce cas on a :

Théorème 5.4 *Soient $G_1 = (V_1, E_1, F_1, F_{v1}, e_1)$ et $G_2 = (V_2, E_2, F_2, F_{v2}, e_2)$ deux graphes plans à trous. Si leurs faces externes sont visibles, alors G_1 et G_2 sont équivalents si et seulement si leurs formes irréductibles sont planes-isomorphes, c'est-à-dire*

$$G_1 \cong G_2 \Leftrightarrow \widehat{G_1} \equiv_p \widehat{G_2}.$$

Preuve : La preuve est identique en tout point à celle du théorème 5.3. Il suffit de remarquer que lorsque les faces externes sont visibles, tant la définition d'équivalence que celle d'isomorphisme plan imposent de les mettre en bijection. □

5.5 Algorithmique des problèmes d'équivalence et d'isomorphisme

5.5.1 Algorithme d'isomorphisme plan

On considère tout d'abord le problème suivant :

Problème : isomorphisme plan de graphes plans à trous (IPGPT)
Instance : deux graphes plans à trous $G = (V, E, F, F_v, e)$ et $G' = (V', E', F', F_v', e')$
Question : G et G' sont-ils plans-isomorphes : $G \equiv_p G'$?

L'objectif de cette partie est de montrer le résultat suivant :

Théorème 5.5 *Le problème* IPGPT *est polynomial (*i.e.*,* IPGPT $\in \mathcal{P}$*). Plus précisément, on peut décider, en temps $\mathcal{O}(|E'| \cdot |E|)$, de l'isomorphisme plan de deux graphes plans à trous $G = (V, E, F, F_v, e)$ et $G' = (V', E', F', F_v', e')$.*

Pour montrer ce théorème, nous allons développer et prouver un algorithme qui résout le problème IPGPT. Le principe est de parcourir les deux graphes en parallèle, en utilisant les arcs (voir la définition 5.4). Pour cela, on définit deux fonctions $succ : A \to A$ et $opp : A \to A$ qui permettent de se déplacer le long des arcs[1]. La fonction *succ* permet, étant donné un arc, d'obtenir le suivant le long de la frontière de la même face, et la fonction *opp* permet, étant donné un arc, d'obtenir l'arc de sens opposé sur la face adjacente. Ainsi, sur la figure 5.xxxi, $succ(a) = b$, $succ(c) = d$, $opp(e) = f$, $opp(a) = g$.

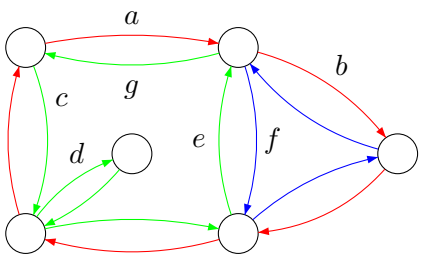


FIG. 5.xxxi – Parcours des arcs : $succ(a) = b$, $succ(c) = d$, $opp(e) = f$, $opp(a) = g$.

Introduire ces deux fonctions est particulièrement intéressant, parce qu'elles permettent de réécrire la définition d'isomorphisme plan sous une forme probablement plus cryptique, mais opérationnelle :

Lemme 5.6 *Soient $G = (V, E, F, F_v, e)$ et $G' = (V', E', F', F_v{}', e')$ deux graphes plans à trous, ayant A et A' pour ensembles respectifs d'arcs. G et G' sont plans-isomorphes si et seulement si il existe une bijection $\rho : A \to A'$ telle que*

1. *ρ commute avec les fonctions succ et opp : $\forall a \in A$*

$$\rho(succ(a)) = succ(\rho(a)) \wedge \rho(opp(a)) = opp(\rho(a)) \ ;$$

2. *ρ préserve les faces visibles et la face externe : $\forall a \in A$*

$$(\underrightarrow{a} \in F_v \text{ si et seulement si } \underrightarrow{\rho(a)} \in F_v{}') \wedge (\underrightarrow{a} = e \text{ si et seulement si } \underrightarrow{\rho(a)} = e').$$

Preuve : ($\Rightarrow$) Comme $G \equiv_p G'$, il existe une bijection $\chi : V \to V'$ d'après la définition 5.9. Pour tout arc $a = xy \in A$, on définit $\rho(a) = \chi(x)\chi(y)$ et les conditions sont vérifiées.

($\Leftarrow$) Tout sommet $v_1 \in V$ est l'extrémité d'une arête $\{v_1, v_2\} \in E$ pour un certain $v_2 \in V$ (car les graphes sont connexes). Donc, v_1 est également l'extrémité d'un arc $v_1v_2 \in A$. En supposant que $\rho(v_1v_2) = v'_1v'_2$, on fixe $\chi(v_1) = v'_1$ et $\chi(v_2) = v'_2$, et on a bien $G \equiv_p G'$. □

[1] Ici, il est important de faire le lien avec les cartes combinatoires introduites dans la définition 1.3 du chapitre 1 : les fonctions *succ* et *opp* correspondent respectivement aux fonctions β_1 et β_2.

Nous pouvons maintenant porter notre attention sur les algorithmes 5 et 6 qui permettent de décider si deux graphes sont plans-isomorphes ou non. Dans l'algorithme 5, on fixe tout d'abord un arc $a_o \in A$, avec $\underrightarrow{a} = e$ (cette restriction n'est pas obligatoire mais permet de réduire le nombre de tests à effectuer). Puis, pour chaque arc $a'_0 \in A'$ tel que $\underrightarrow{a'_0} = e'$ (la face externe doit être préservée), on appelle l'algorithme 6 afin de construire une fonction $f : A \to A'$. Ensuite, si f satisfait les conditions du lemme 5.6, alors l'algorithme renvoie VRAI. Et si aucune fonction f ne satisfaisant les conditions du lemme n'est trouvée, alors l'algorithme renvoie FAUX. Quant à l'algorithme 6, il permet de parcourir les graphes, en commençant par a_0 et a'_0. Les fonctions *succ* et *opp* sont utilisées l'une après l'autre pour découvrir de nouveaux arcs. À chaque découverte d'un nouvel arc $a_1 \in A$ à partir d'un arc $a \in A$ avec la fonction *succ* (respectivement *opp*), la valeur de $f[a_1]$ est l'arc $succ(f[a])$ (respectivement $opp(f[a])$).

Algorithme 5: VERIFIERISOMORPHISMEPLAN(G, G')

Données : deux graphes plans à trous $G = (V, E, F, F_v, e)$ et $G' = (V', E', F', F_v', e')$ ayant respectivement A et A' comme ensembles d'arcs

Résultat : VRAI si $G \equiv_p G'$, FAUX sinon

```
1 choisir a_0 ∈ A tel que a_0→ = e ;
2 pour chaque a'_0 ∈ A' tel que a'_0→ = e' faire
3     f ← PARCOURIRETCONSTRUIREAPPARIEMENT(G, G', a_0, a'_0) ;
4     si f satisfait les conditions du lemme 5.6 alors
5         retourner VRAI
6 retourner FAUX
```

Preuve : (preuve du théorème 5.5) Si VERIFIERISOMORPHISMEPLAN retourne VRAI, alors il existe $f : A \to A'$ qui satisfait les conditions du lemme 5.6, et par conséquent $G \equiv_p G'$.

Réciproquement, supposons que $G \equiv_p G'$. Alors il existe une fonction $\rho : A \to A'$ qui satisfait les conditions du lemme 5.6. Soit a_0 l'arc choisi à la ligne 1 de l'algorithme VERIFIERISOMORPHISMEPLAN ($\underrightarrow{a_0} = e$). Comme la boucle des lignes 2 à 5 parcourt tous les arcs de la face externe de G', il existe une itération pour laquelle $a'_0 = \rho(a_0)$. Nous affirmons alors que pour cette itération précise, PARCOURIRETCONSTRUIREAPPARIEMENT retourne f telle que $\forall a \in A$, $f[a] = \rho(a)$. En effet, on a les deux propriétés suivantes :

1. *Lorsque l'on empile un arc a dans P,* $f[a] = \rho(a)$. On le montre par induction. Initialement (ligne 5), on a bien $f[a_0] = a'_0 = \rho(a_0)$ (du fait de la ligne 3). Cela est également vrai pour la ligne 10 puisque la valeur de $f[succ(a)]$ est fixée à $succ(f[a])$ (en ligne 9), et à la ligne 13 puisque $f[opp(a)] = opp(f[a])$ (ligne 12). Or par hypothèse d'induction, $f[a] = \rho(a)$, donc comme $(\rho(a) = a' \Rightarrow \rho(succ(a)) =$

Algorithme 6: PARCOURIRETCONSTRUIREAPPARIEMENT(G, G', a_0, a'_0)

Données : deux graphes plans à trous $G = (V, E, F, F_v, e)$ et $G' = (V', E', F', F_v', e')$, et deux arcs $a_0 \in A$ et $a'_0 \in A'$
Résultat : un tableau $f : A \to A'$

```
1  pour chaque a ∈ A faire
2      f[a] ← nil
3  f[a_0] ← a'_0 ;
4  soit P une pile vide ;
5  empiler a_0 dans P ;
6  tant que P n'est pas vide faire
7      dépiler un arc a de P ;
8      si f[succ(a)] = nil alors
9          f[succ(a)] ← succ(f[a]) ;
10         empiler succ(a) dans P
11     si f[opp(a)] = nil alors
12         f[opp(a)] ← opp(f[a]) ;
13         empiler opp(a) dans P
14 retourner f
```

$succ(a'))$ et $(\rho(a) = a' \Rightarrow \rho(opp(a)) = opp(a'))$, par les conditions du lemme 5.6, on a bien $f[succ(a)] = \rho(succ(a))$ et $f[opp(a)] = \rho(opp(a))$.

2. *Tout arc $a \in A$ est empilé une et une seule fois dans P.* En effet, comme nous ne considérons que des graphes connexes, tout sommet a est accessible depuis a_0 par au moins une chaîne $a_0, \ldots, a_n$ telle que $a_n = a$ et, $\forall k \in \{1, 2, \ldots, n\}$, soit $a_k = succ(a_{k-1})$, soit $a_k = opp(a_{k-1})$. Ainsi, à chaque fois qu'un des arcs a_i est dépilé de P (ligne 7), a_{i+1} est empilé dans P (lignes 10 et 13) s'il n'a pas déjà été découvert par un autre chemin (lignes 8 et 11).

Par conséquent, VERIFIERISOMORPHISMEPLAN retourne VRAI.

Concernant la complexité, l'algorithme PARCOURIRETCONSTRUIREAPPARIEMENT est en $\mathcal{O}(|A|)$: la boucle de la ligne 6 est répétée $|A|$ fois car, d'une part, exactement un arc est dépilé à chaque itération, et, d'autre part, chaque arc $a \in A$ est empilé une et une seule fois (tests des lignes 8 et 11, suivis des assignations des lignes 9 et 12). Concernant l'algorithme VERIFIERISOMORPHISMEPLAN, le test de la ligne 4 peut être réalisé en $\mathcal{O}(|A|)$ (le lemme 5.6 consistant uniquement à vérifier tous les arcs, et le statut des faces adjacentes). La boucle de la ligne 2 est répétée au plus $|A'|$ fois. Finalement, la complexité globale de l'algorithme 5 est en $\mathcal{O}(|A'| \cdot |A|)$, ou encore en $\mathcal{O}(|E'| \cdot |E|)$. □

Remarquons qu'il peut exister des algorithmes plus efficaces : nous montrons que le problème est polynomial, mais nous ne donnons certainement pas d'algorithme optimal. Par exemple, il est possible de faire appel à des informations topologiques, qui sont

validées par des données géométriques [Sommellier, 1997].

5.5.2 Algorithme d'isomorphisme sphérique

On aborde maintenant le problème de l'isomorphisme sphérique :

Problème : isomorphisme sphérique de graphes plans à trous (ISGPT)
Instance : deux graphes plans à trous $G = (V, E, F, F_v, e)$ et $G' = (V', E', F', F_v', e')$
Question : G et G' sont-ils sphères-isomorphes : $G \equiv_s G'$?

On souhaite démontrer le résultat suivant :

Théorème 5.6 *Le problème* ISGPT *est polynomial. Plus précisément, on peut décider, en temps* $\mathcal{O}(|E'| \cdot |E|)$*, de l'isomorphisme sphérique de deux graphes plans à trous* $G = (V, E, F, F_v, e)$ *et* $G' = (V', E', F', F_v', e')$.

Ce résultat découle directement de ceux établis dans la partie 5.5.1 : le lemme 5.6, sans la condition sur les faces externes, reste vrai ; par conséquent il suffit de réutiliser l'algorithme 5 en omettant, ligne 4, le test de préservation des faces externes.

5.5.3 Algorithme d'équivalence

Pour finir, nous nous intéressons au problème suivant :

Problème : équivalence de graphes plans à trous (EGPT)
Instance : deux graphes plans à trous $G = (V, E, F, F_v, e)$ et $G' = (V', E', F', F_v', e')$
Question : G et G' sont-ils équivalents : $G \cong G'$?

On obtient alors le résultat suivant, qui est, à notre sens, la principale contribution de ce chapitre :

Théorème 5.7 *Le problème* EGPT *est polynomial.*

Preuve : Décrivons la procédure qui permet de résoudre ce problème. La première étape consiste à vérifier le statut des faces externes. En effet, si ces dernières n'ont pas le même statut, alors on peut d'ores et déjà conclure que les graphes ne sont pas équivalents. Dans le cas contraire, on normalise les graphes en utilisant l'algorithme 1 afin d'obtenir leurs formes irréductibles. Ensuite, on peut distinguer deux cas. Si les deux faces externes sont invisibles, alors d'après le théorème 5.3, les deux graphes sont équivalents si et seulement si leurs formes irréductibles sont sphères-isomorphes, ce que l'on sait décider en temps polynomial d'après le théorème 5.6. Si les deux faces externes sont visibles, alors c'est le théorème 5.4 qui s'applique, et les deux graphes sont équivalents si et seulement si leurs formes irréductibles sont planes-isomorphes, ce qui est également décidable en temps polynomial d'après le théorème 5.5. □

5.6 Recherche de motifs dans les graphes plans à trous

Nous nous intéressons maintenant à l'isomorphisme plan de sous-graphe : il s'agit de retrouver un motif dans une image.

5.6.1 Définition

Commençons par définir un motif.

Définition 5.17 (Motif)
Soient M et $G = (V, E, F, F_v, e)$ deux graphes plans à trous. On dit que M est un motif *(ou sous-graphe) de G s'il existe un graphe $G' = (V, E, F, F_v{}', e)$ tel que*

1. *$F_v{}' \subseteq F_v$;*
2. *les faces de $F_v{}'$ sont contiguës ;*
3. *$M \equiv_p \widehat{G'}$.*

En d'autres termes, un motif s'obtient en rendant certaines faces du graphe d'origine invisibles, puis en normalisant. Le raisonnement pour obtenir un motif est une nouvelle fois intimement lié aux faces, et non plus aux sommets et/ou aux arêtes, comme c'est le cas dans les définitions classiques des sous-graphes (voir la partie 3.1.1).

Cette définition a plusieurs conséquences. Tout d'abord, un motif est nécessairement irréductible. En conséquence, il s'avère qu'un motif M peut avoir une taille plus importante que le graphe G dont il est issu, c'est-à-dire qu'on peut avoir $|M| > |G|$, suite à la normalisation. C'est le cas sur la figure 5.xxxii, puisqu'il existait une charnière au niveau des pattes du manchot. Enfin, tous les motifs ne sont pas possibles car les faces visibles doivent rester contiguës.

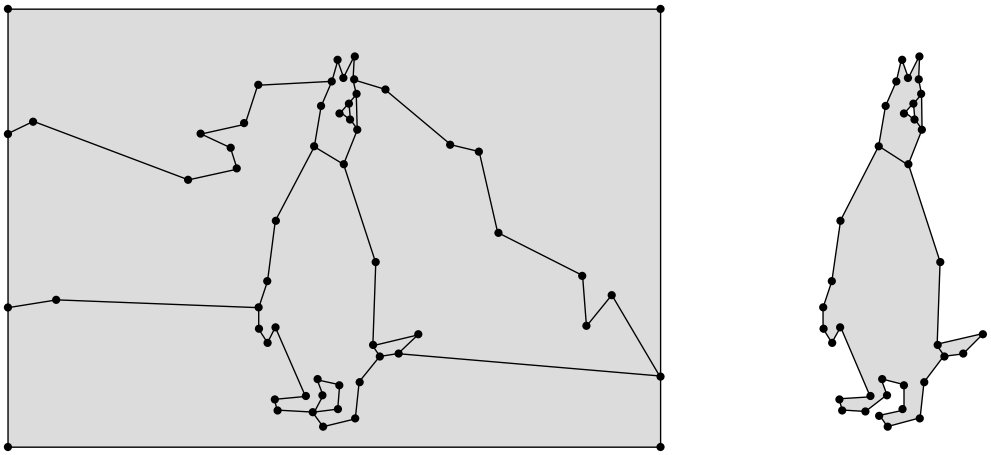


FIG. 5.xxxii – Un graphe plan à trous (à gauche) et un motif possible (à droite).

5.6.2 Algorithme de recherche de motifs

On considère le problème suivant :

Problème : motif d'un graphe plan à trous (MGPT) **Instance :** deux graphes plans à trous $M = (V, E, F, F_v, e)$ et $G = (V', E', F', F_v', e')$ **Question :** M est-il un motif de G ?

Dans cette partie, on montre :

Théorème 5.8 *Le problème* MGPT *est polynomial. Plus précisément, on peut décider, en temps* $\mathcal{O}(|E'| \cdot |E|)$*, si le graphe plan à trous* $M = (V, E, F, F_v, e)$ *est un motif de* $G = (V', E', F', F_v', e')$*.*

Pour montrer ce théorème, on utilise l'algorithme 7 qui se base sur la définition 5.17. Il consiste à :

1. mettre en correspondance les faces visibles de M avec des faces visibles de G. C'est le but de l'algorithme 8 sur lequel nous reviendrons ;
2. disposant des faces visibles candidates dans G, on normalise G par rapport à celles-ci, de façon à obtenir le graphe noté $\widehat{G'}$ de la définition 5.17 ;
3. il ne reste plus qu'à vérifier que M et $\widehat{G'}$ sont plans-isomorphes.

Algorithme 7: VERIFIERMOTIF(M, G)

Données : deux graphes plans à trous $M = (V, E, F, F_v, e)$ et $G = (V', E', F', F_v', e')$ ayant respectivement A et A' comme ensembles d'arcs
Résultat : VRAI si M est un motif de G, FAUX sinon

```
1  choisir a_0 ∈ A tel que a_0 ∈ F_v ;
2  pour chaque a'_0 ∈ A' tel que a'_0 ∈ F_v' faire
3      ρ ← PARCOURIRFACESVISIBLESETCONSTRUIREAPPARIEMENT(M, G, a_0, a'_0) ;
4      F_v'' ← ∅ ;
5      pour chaque a ∈ A tel que a ∈ F_v faire
6          F_v'' ← F_v'' ∪ {ρ[a]}
7      si F_v'' ⊆ F_v' et F_v'' est un ensemble de faces contiguës alors
8          G'' ← (V', E', F', F_v'', e') ;
9          G'' ← NORMALISERGRAPHE(G'') ;
10         si VERIFIERISOMORPHISMEPLAN(M, G'') alors
11             retourner VRAI
12 retourner FAUX
```

Concernant le premier point, visant à mettre en correspondance les faces visibles de M avec certaines de G, il est naturel de passer par les arcs, comme on le fait dans l'algorithme 8. Ce dernier fonctionne de façon analogue à la procédure 6 mise en œuvre dans le cadre du test de l'isomorphisme plan. Notons d'ailleurs que ces deux algorithmes sont très similaires. En effet, seules les lignes 11 changent, car le premier ne considère que les faces visibles, alors que le second porte sur toutes les faces. Néanmoins, comme la fonction que l'on récupère concerne uniquement des arcs délimitant des faces visibles, elle nous permet ensuite, dans l'algorithme 7 lignes 4-6, de les apparier entre elles.

Quant aux deux derniers points de normalisation et de test d'isomorphisme plan, on réutilise les résultats détaillés dans les parties 5.4.3 et 5.5.1. Il va de soit ici que de nombreuses optimisations pourraient être apportées dans la mesure où la fonction f retournée par l'algorithme 8 pourraient probablement être directement exploitée par VERIFIERISOMORPHISMEPLAN. Cependant, il conviendrait d'une part d'être vigilants sur les conditions devant être vérifiées par f, puisqu'en l'état, f ne donne aucune information sur les faces invisibles. D'autre part, la classe de complexité du problème resterait inchangée : il est polynomial, ce qu'on souhaitait démontrer.

Algorithme 8: PARCOURIRFACESVISIBLESETCONSTRUIREAPPARIEMENT(G, G', a_0, a'_0)

Données : deux graphes plans à trous $G = (V, E, F, F_v, e)$ et $G' = (V', E', F', F_v', e')$, et deux arcs $a_0 \in A$ et $a'_0 \in A'$ tels que $\underrightarrow{a_0} \in F_v$ et $\underrightarrow{a'_0} \in F_v'$

Résultat : un tableau $f : A \to A'$

1 **pour chaque** $a \in A$ **faire**
2 $f[a] \leftarrow nil$
3 $f[a_0] \leftarrow a'_0$;
4 soit P une pile vide ;
5 empiler a_0 dans P ;
6 **tant que** P *n'est pas vide* **faire**
7 dépiler un arc a de P ;
8 **si** $f[succ(a)] = nil$ **alors**
9 $f[succ(a)] \leftarrow succ(f[a])$;
10 empiler $succ(a)$ dans P
11 **si** $f[opp(a)] = nil$ *et* $\underrightarrow{opp(a)} \in F_v$ **alors**
12 $f[opp(a)] \leftarrow opp(f[a])$;
13 empiler $opp(a)$ dans P
14 **retourner** f

5.6.3 Expérimentations préliminaires

Dans la mesure où les résultats de ce chapitre sont relativement récents, cette partie présente plusieurs expérimentations préliminaires portant sur la recherche de motifs

dans des images. Il conviendrait de les compléter par la suite.

À chaque fois, les graphes utilisés sont plans et correspondent à la définition des graphes plans à trous que nous avons présentée. Cependant, pour des questions pratiques, $F \backslash F_v = \emptyset$. En effet, une recherche de motifs à trous impose de spécifier au préalable les faces visibles. Or, l'introduction de faces invisibles devrait se faire manuellement pour s'assurer que les motifs recherchés aient un sens. Cependant, cette façon de procéder est incompatible avec des expérimentations à grande échelle. Néanmoins, fondamentalement, les algorithmes sont les mêmes, et seule la phase de normalisation devient inutile.

Graphes plans et cartes combinatoires

Les expérimentations de cette section proviennent de travaux ayant porté, en première intention, sur les cartes combinatoires. Les résultats que nous présentons ont donc été obtenus sur les cartes combinatoires (nous préciserons les protocoles par la suite) et non sur les graphes. Cependant, ces dernières, que nous avons préalablement définies (définition 1.3), peuvent être vues comme une structure de données permettant de représenter des graphes plans (c'est alors une alternative aux matrices d'adjacence, ou encore aux tableaux de listes d'adjacence). À cet effet, on ne stocke qu'un triplet $M = (D, \beta_1, \beta_2)$ qui contient un ensemble de brins (*darts*) D qui correspondent aux arcs que nous avons utilisés pour le parcours des faces. β_1 est une permutation permettant, étant donné un brin, d'obtenir le brin suivant le long de la frontière d'une face. C'est l'équivalent de notre fonction *succ*(). β_2 est, elle, une involution retournant, étant donné un brin, le brin opposé. Elle permet donc de passer d'une face à une autre, de façon analogue à la fonction *opp*() que nous avons utilisée. La condition *1.* du lemme 5.6 se ramène ainsi à trouver une fonction $f : D \rightarrow D'$ telle que $\forall d \in D$, $\forall i \in \{1, 2\}$, $f(\beta_i(d)) = \beta'_i(f(d))$.

Cependant, dans une carte combinatoire, aucune face ne joue le rôle de face externe, et c'est donc l'isomorphisme sphérique qui est résolu. Pour pallier cette caractéristique, on autorise β_2 à être partiellement définie (on parle de *carte combinatoire ouverte*) en introduisant un nouvel élément dans l'ensemble des brins, ϵ. Les arcs opposés à la face externe sont alors liés à ϵ par β_2 (toute face du graphe peut donc être associée à une face de la carte, à l'exception de la face externe).

Recherche de motifs dans des images

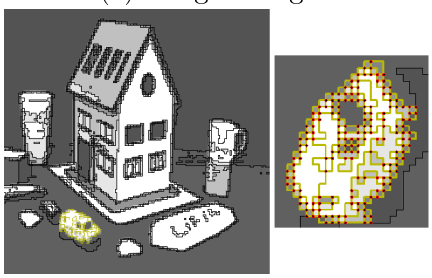
Une première expérimentation a consisté en la recherche de motifs dans des images. Pour cela, nous avons considéré la base d'images MOVI [Luo et al., 2003], qui est constituée d'images 2D d'objets 3D (des maisons parfois entourées de divers objets, voir la partie 2.3.3 page 57 pour une illustration). Pour chaque maison, plusieurs images, prises suivant différents points de vue, sont disponibles.

Nous avons utilisé deux approches pour ces expérimentations. Premièrement, nous avons extrait directement des images des cartes combinatoires, selon la méthode décrite dans [Damiand et al., 2004]. La seconde approche consista à utiliser les points d'intérêts

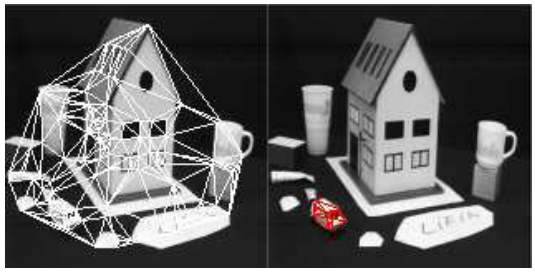
fournis avec les images afin de construire des graphes plans, par une triangulation de Delaunay, puis de les convertir en cartes combinatoires.

Quelque soit l'approche considérée, nous avons ensuite extrait des images des motifs, puis utilisé nos algorithmes pour les rechercher. La figure 5.xxxiii fournit un exemple : en haut (ligne (a)) se trouve l'image d'origine ; le motif recherché correspond à la voiture en bas à gauche de la maison. La ligne (b) montre, à gauche, la carte combinatoire correspondant à l'image entière (8114 brins et 1700 faces) et, à droite, celle du motif seul (362 brins et 132 faces). Au cours des expérimentations, le motif a été retrouvé dans l'image d'origine, en 60ms, après avoir été soumis à une rotation. La dernière ligne concerne le graphe de Delaunay des points d'intérêt (140 sommets, 404 arêtes et 266 faces), et le sous-graphe correspondant au motif (16 sommets, 38 arêtes et 23 faces). Une nouvelle fois, ce motif a été retrouvé dans l'image d'origine, et en 10ms.

(a) Image d'origine



(b) À gauche : carte combinatoire ; à droite : motif recherché



(c) À gauche : graphe de Delaunay des points d'intérêt ; à droite : motif recherché

FIG. 5.xxxiii – Recherche de motifs dans des images.

Les expérimentations menées sur la base montrent que les motifs ont toujours été retrouvés dans les images dont ils avaient été extraits, et uniquement dans celles-ci, nos

	$sg_{i,5\%}$		$sg_{i,10\%}$		$sg_{i,20\%}$		$sg_{i,33\%}$		$sg_{i,50\%}$	
g_i	vf2	map	vf2	map	vf2	map	vf2	map	vf2	map
g_{500}	0.08	0.07	0.04	0.10	0.47	0.03	0.7	0.02	10.4	0.10
g_{1000}	4.7	0.21	2.54	0.07	0.55	0.05	7.31	0.06	12.7	0.06
g_{5000}	12.3	0.28	156.5	0.31	> 3600	0.31	> 3600	0.31	> 3600	0.31

TAB. 5.1 – Résultats des expérimentations sur le passage à l'échelle. Dans chaque cellule se trouvent les temps de résolution du problème, en seconde, par Vflib2 (vf2) et notre approche (map).

algorithmes recherchant des isomorphismes non tolérants. Notons que nous n'avons pas non plus eu de faux-positifs, ce qui peut être le cas lorsque les motifs recherchés sont de taille trop petite.

Passage à l'échelle

La seconde série d'expérimentations a consisté à étudier les propriétés de passage à l'échelle des algorithmes, et à les comparer avec l'état de l'art.

Pour cela, nous avons généré aléatoirement trois graphes plans g_{500}, g_{1000} de taille 500, 1000 et 5000. Ils ont été obtenus en choisissant aléatoirement n points dans le plan, puis en reliant ces points par une triangulation de Delaunay. Pour chaque graphe g_i, nous avons ensuite généré 5 sous-graphes notés $sg_{i,k\%}$, qui contiennent $k\%$ des sommets de g_i, avec $k \in \{5,\ 10,\ 20,\ 33,\ 50\}$.

Notre algorithme a été comparé à Vflib2 [Cordella et al., 2001], approche permettant de résoudre le problème de l'isomorphisme de sous-graphes (nous ne présentons que cet algorithme qui, dans nos expérimentations, a toujours été plus rapide que Vflib et Ullman[2]). Le tableau 5.1 récapitule les résultats obtenus : il est vite plus efficace d'utiliser nos algorithmes lorsque les graphes ont une taille importante. Notons en particulier que pour le graphe g_{5000}, nous résolvons toujours le problème en moins de 1 seconde, alors que Vflib2 ne donne toujours aucun résultat après 1h.

Il faut cependant noter que notre algorithme ne résout pas exactement le même problème que Vflib2. En effet, ce dernier recherche des sous-graphes induits (voir la définition 3.3 de la partie 3.1.1), c'est-à-dire que le sous-graphe est défini par ses sommets, et que toutes les arêtes adjacentes en font partie. Pour notre part, comme nous raisonnons sur les faces, nous n'obtenons pas forcément des sous-graphes induits. De plus, nous tirons parti du fait que les graphes soient plans, tandis que Vflib résout le problème sur les graphes généraux. Une conséquence est que le nombre de résultats trouvés peut ne pas être le même (Vflib peut ne pas trouver un sous-graphe car il lui manque des arêtes pour être induit, mais il peut également trouver un sous-graphe que nous ne trouvons pas car l'organisation des faces est différente). Par exemple, $sg5000, 10\%$ a été trouvé deux fois par Vflib, et seulement une fois par notre approche.

[2]Ces algorithmes font partie de la librairie *VFLib Graph Matching* http://amalfi.dis.unina.it/graph/vflib-2.0/doc/vflib.html.

5.7 Discussion

Dans cette partie, nous commençons par nous positionner par rapport aux travaux de Cori (1975) sur les cartes combinatoires. Nous présenterons ses définitions et son processus de numérotation des brins d'une carte, avant de nous intéresser aux points communs et aux différences avec notre approche. Dans un second temps, nous revenons sur le cas des graphes plans non connexes et tentons d'esquisser une liste des difficultés qu'ils ajoutent.

5.7.1 Positionnement par rapport aux travaux de Cori (1975)

Dans cette partie, nous nous positionnons par rapport aux travaux de Robert Cori, portant notamment sur l'isomorphisme de cartes combinatoires. Il les a publiés dans sa thèse [Cori, 1973], qui parut ensuite dans un numéro de la collection Astérisque de la Société Mathématique de France [Cori, 1975]. Les numéros de pages que nous donnons proviennent de cette dernière référence.

Commençons par donner quelques définitions. Page 12, l'auteur définit les hypercartes comme suit :

Définition 5.18 (Hypercarte)
Une hypercarte *est un couple* (σ, α) *de permutations opérant sur un ensemble fini* B *(ensemble de brins), tel que le groupe engendré par* $\{\sigma, \alpha\}$ *opère transitivement sur* B.

Ajoutons qu'une hypercarte est dite *pointée* si on distingue un élément b^* de l'ensemble des brins. En outre, il s'avère que, sous une condition particulière, les hypercartes deviennent des cartes. Ainsi, page 13, on a :

Définition 5.19 (Carte)
Une hypercarte (σ, α) *est une* carte *si* α *est une involution sans point fixe.*

Ceci signifie que α ne contient que des cycles de longueur 2. Par conséquent, cette définition est identique à celle de carte combinatoire 2D de [Lienhardt, 1991], donnée page 30. De plus, par rapport aux notations que nous utilisons dans [Damiand et al., 2009b, Damiand et al., 2011], σ correspond à β_1, et α à β_2. Enfin, dans cette thèse, σ est la fonction *succ*() sur les arcs, et α la fonction *opp*().

Cori propose ensuite une définition de l'isomorphisme d'hypercartes page 28 :

Définition 5.20 (Isomorphisme de deux hypercartes)
Deux hypercartes (σ_1, α_1) *opérant sur* B_1 *et* (σ_2, α_2) *opérant sur* B_2 *sont dites isomorphes s'il existe une bijection* λ *entre* B_1 *et* B_2, *telle que* $\alpha_1 = \lambda^{-1}\alpha_2\lambda$ *et* $\sigma_1 = \lambda^{-1}\sigma_2\lambda$.

Une nouvelle fois, restreinte aux cartes, cette définition est similaire à celle de [Lienhardt, 1994], que nous avons utilisée dans notre article [Damiand et al., 2009b] :

Définition 5.21 (Isomorphisme de cartes combinatoires 2D)
Deux cartes combinatoires $M = (D, \beta_1, \beta_2)$ *et* $M' = (D', \beta'_1, \beta'_2)$ *sont isomorphes s'il existe une bijection* $f : D \to D'$, *telle que* $\forall d \in D$, $\forall i \in \{1, 2\}$, $f(\beta_i(d)) = \beta'_i(f(d))$.

Cette définition est également proche de celle que nous donnons page 126 pour les graphes plans à trous, comme nous l'établissons dans le lemme 5.6 page 143.

Notons que deux hypercartes pointées $(\sigma_1, \alpha_1, b_1^*)$ et $(\sigma_2, \alpha_2, b_2^*)$ sont isomorphes si elles sont isomorphes en tant qu'hypercartes, et si de plus $\lambda b_1^* = b_2^*$.

Par la suite, pages 41-42, Cori donne un processus de numérotation ζ des brins d'une hypercarte. Ce processus, qui vise à associer un numéro unique à chaque brin, est fondamental dans sa thèse, puisqu'elle porte sur l'étude de ces codes. Il permet également de résoudre le problème d'isomorphisme de cartes, et il s'avère que nous avons suivi la même idée, comme nous allons le voir.

Concernant le processus de numérotation de Cori, il est décrit de la manière suivante :

- b_0 est choisi quelconque (si l'hypercarte est pointée par b^*, alors $b_0 = b^*$) ;
- supposons définis b_0, $b_1 = \zeta b_0$, $b_2 = \zeta^2 b_0$, ..., $b_p = \zeta^p b_0$, on définit $\zeta b_p = b_{p+1}$ comme suit :
  - Procédure A : - Si αb_p appartient à un sommet (cycle de σ) sur lequel aucune brin n'appartient à $\{b_0, b_1, b_2, \ldots, b_p\}$, alors $\zeta b_p = \alpha b_p$.
  - Procédure B : - Si αb_p appartient à un sommet sur lequel un brin appartient à $\{b_0, b_1, b_2, \ldots, b_p\}$, alors $\zeta b_p = \sigma b_p$ si σb_p n'appartient pas à $\{b_0, b_1, b_2, \ldots, b_p\}$, sinon :
  - Procédure C : - ζb_p est le premier élément de la suite $\sigma b_{p-1}, \sigma b_{p-2}, \sigma b_{p-3}, \ldots, \sigma b_2, \sigma b_1, \sigma b_0$ qui n'appartient pas à $\{b_0, b_1, b_2, \ldots, b_p\}$.

Un exemple de résultat obtenu est représenté sur la figure 5.xxxiv.

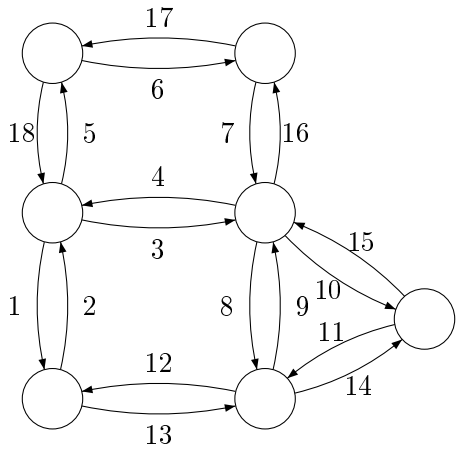


Fig. 5.xxxiv – Numérotation de Cori.

De la même façon, l'algorithme que nous avons donné page 145 peut-être réécrit pour numéroter les arcs. Il s'agit de l'algorithme 9 : NUMEROTATIONARCS(G, a_0). Dans ce cas, on obtient la numérotation de la figure 5.xxxv.

Au-delà de la différence de numérotation qu'on obtient, il convient de remarquer que

Algorithme 9: NUMEROTATIONARCS(G, a_0)

Données : un graphe plan à trous $G = (V, E, F, F_v, e)$ et un arc $a_0 \in A$
Résultat : un tableau $\zeta : A \to \{1, 2, \ldots, |A|\}$, numérotation des arcs de G

```
1  pour chaque a ∈ A faire
2      ζ[a] ← 0
3  ζ[a_0] ← 1 ;
4  i ← 2 ;
5  soit P une pile vide ;
6  empiler a_0 dans P ;
7  tant que P n'est pas vide faire
8      dépiler un arc a de P ;
9      si ζ[succ(a)] = 0 alors
10         ζ[succ(a)] ← i ;
11         i ← i + 1 ;
12         empiler succ(a) dans P
13     si ζ[opp(a)] = 0 alors
14         ζ[opp(a)] ← i ;
15         i ← i + 1 ;
16         empiler opp(a) dans P
17 retourner ζ
```

notre approche conduit à un algorithme linéaire en le nombre d'arcs. Or, si le processus de numérotation de Cori n'est pas suffisamment précis pour établir une complexité (dans la mesure où aucune structure de données n'est précisée), le fait de vérifier si un brin appartient à un cycle de σ dont un brin a déjà été numéroté, dans les procédures A et B, sous-entend que l'algorithme est quadratique.

En terme d'isomorphisme, maintenant, l'auteur conclut, page 44, en indiquant que, étant donné $\tilde{\alpha} = \lambda^{-1}\alpha\lambda$ et $\tilde{\sigma} = \lambda^{-1}\sigma\lambda$, et $\lambda b = 1$, alors $(\tilde{\sigma}, \tilde{\alpha}, 1)$ et (σ, α, b) ont la même numérotation. Ceci signifie donc que deux hypercartes sont isomorphes si et seulement si on peut les exprimer sous la forme de deux hypercartes pointées ayant une numérotation identique.

C'est la même conclusion à laquelle nous arrivons avec notre algorithme : de fait, nous avons suivi la même idée (Cori a d'ailleurs étendu ses travaux en proposant un algorithme d'automorphisme de cartes, que l'on pourrait étendre à l'isomorphisme de cartes [Cori, 1985]). Néanmoins, il faut remarquer que les motivations du travail de Cori et les nôtres sont différentes. En effet, par la suite, l'étude de Cori se poursuit sur les propriétés combinatoires et analytiques de sa numérotation. De notre côté, nous obtenons un algorithme sur les graphes plans à trous (qu'on peut difficilement modéliser avec des cartes [Damiand et al., 2011]) et qui nous permet également d'attaquer l'isomorphisme de sous-graphe, qui n'est pas abordé par Cori.

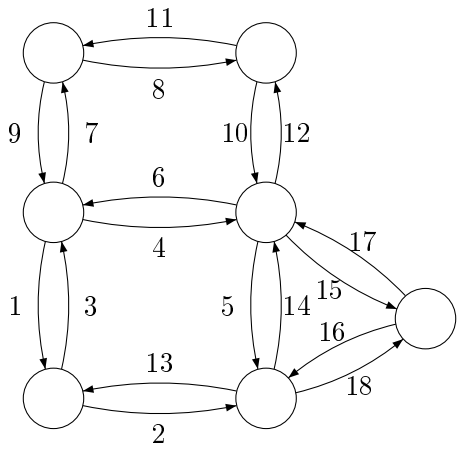


FIG. 5.xxxv – Notre numérotation.

5.7.2 Retour sur la connexité

Tout au long de ce chapitre, nous avons exigé des graphes plans à trous qu'ils soient connexes, et que leurs faces visibles soient contiguës. Or, comme nous l'avons annoncé, lever ces hypothèses soulève de nombreux problèmes (et perspectives de recherche) que nous souhaitons développer maintenant.

Considérons donc le plongement d'un graphe planaire non connexe comme celui de la figure 5.xxxvi. Clairement, plusieurs de ses faces comme f_1 (ou f_2) n'ont plus une frontière unique, mais plusieurs ; on parle alors de faces *composées*. Plus précisément, f_1 a deux frontières qui sont $[v_1v_2v_3v_4]$ et $[v_{10}v_{11}v_{12}]$. En conséquence, on ne peut plus décrire un tel plongement avec les SDGP de la définition 5.3. Clairement, d'un point de vue syntaxique, on peut facilement étendre cette définition. Mais d'un point de vue sémantique, par contre, nous ne connaissons pas actuellement les conditions nécessaires permettant d'établir un analogue à la conjecture 5.2.3 permettant de montrer qu'un SDGP correspond bien à un graphe plan. Dit autrement, nous ne savons pas définir les SDGP *bien fondés* dans ce cadre.

Concernant maintenant l'isomorphisme plan, nous savons par contre qu'il est polynomial. Cependant, la procédure est plus sophistiquée que celle que nous avons présentée dans ce chapitre : elle consiste à tester l'isomorphisme entre les composantes connexes de chaque graphe, puis à vérifier que chaque composante du premier graphe est bien appariée à une unique composante du second et réciproquement. Cependant, la difficulté de la preuve vient de la notion même de composante connexe, dans la mesure où ces composantes sont en elles-mêmes des graphes plans à trous, munis de faces externes, et décrits par des SDGP. Ainsi, c'est l'algorithme de décomposition d'un SDGP en composantes connexes qui n'est pas trivial.

Concernant l'équivalence, utiliser des graphes non connexes pose également des dif-

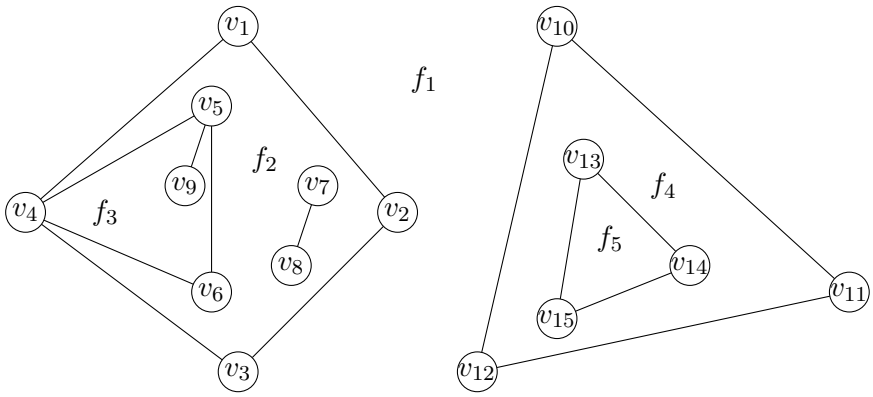


FIG. 5.xxxvi – Un graphe ayant des faces composées.

ficultés. En particulier, si la procédure de normalisation que nous avons décrite dans la partie 5.4.3 peut s'étendre, le théorème 5.4 stipulant que deux graphes plans équivalents ont des formes irréductibles planes-isomorphes n'est plus vrai, comme le montrent les figures 5.xxxvii et 5.xxxviii. En effet, sur chaque image est représenté un graphe plan à trous, dont les faces externes sont visibles. Ces deux graphes sont équivalents (on ne considère alors que les faces visibles). Cependant, leurs formes normales ne sont pas planes-isomorphes.

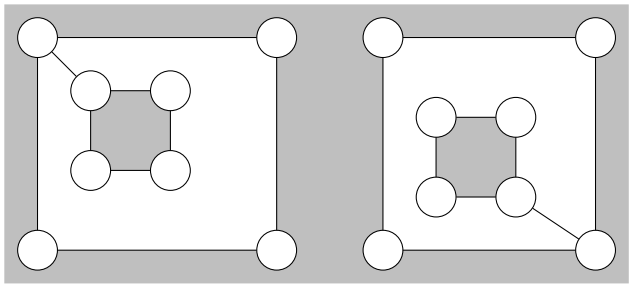


FIG. 5.xxxvii – Graphe plan à trous non connexe, dont la face externe est visible. Il est équivalent au graphe de la figure 5.xxxviii, mais leurs formes irréductibles ne sont pas planes-isomorphes.

Ainsi, pour faire le lien entre équivalence et isomorphisme plan, il ne faut pas utiliser les formes irréductibles telles que nous les avons définies, mais passer, de nouveau, par leurs composantes connexes.

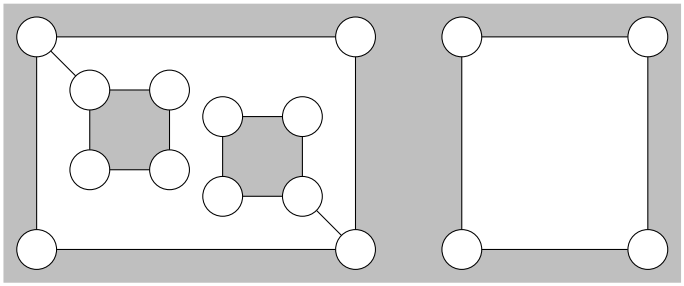


FIG. 5.xxxviii – Graphe plan à trous non connexe, dont la face externe est visible. Il est équivalent au graphe de la figure 5.xxxvii, mais leurs formes irréductibles ne sont pas planes-isomorphes.

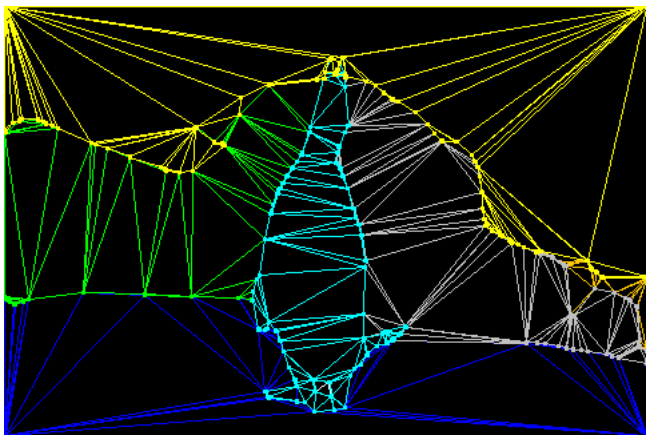
Enfin, la recherche de motifs pose elle aussi de nouvelles difficultés. En effet, si la recherche de motifs connexes dans des graphes plans non connexes est une extension des résultats précédents, nous conjecturons en revanche que le problème est $\mathcal{NP}$-complet lorsque les motifs eux-mêmes ne sont pas connexes.

5.8 Conclusion

Dans ce chapitre, nous avons défini une syntaxe cohérente et complète permettant de manipuler les graphes plans à trous. En outre, nous avons procédé à une distinction fine entre isomorphisme classique, sphérique et plan. De plus, nous nous sommes attachés à définir l'équivalence de graphes plans, et avons mis en place une procédure de normalisation permettant de mettre au jour les liens existant entre isomorphismes et équivalence. Enfin, nous avons proposé des algorithmes polynomiaux permettant de résoudre, dans le cadre des graphes plans à trous, les problèmes d'isomorphismes, d'équivalence, et de recherche de motifs.

Conclusion et perspectives

Ceci non plus n'est pas un manchot



Mais ceci est un graphe plan, et nous avons désormais les moyens de l'extraire automatiquement de toute image d'une part, et d'y appliquer un algorithme polynomial d'isomorphisme ou de recherche de motif d'autre part.

Notre démarche consistait initialement à définir une fonction de coût sur les appariements de graphes issus d'images, permettant de distinguer un bon appariement d'un mauvais. Dans l'idéal, nous aurions alors disposé de critères pour la mise en place d'une méthode d'optimisation d'appariements pouvant porter ses fruits, notamment en classification d'images.

Les résultats présentés dans le chapitre 2 sont en demi-teinte, et indiquent que nos tentatives n'ont pas abouti. En effet, ils ne nous ont pas permis de mettre au jour des critères significatifs, bien que le respect de la structure du graphe puisse être dans certains cas discriminant. Ceci est sans nul doute dû au fait que notre approche privilégie le graphe seul, en faisant abstraction des spécificités du domaine de l'image. Cela constituait cependant une volonté de notre part, afin de ne pas masquer le graphe, qui reste notre sujet d'étude privilégié.

En conséquence, nous avons reconsidéré nos objectifs, qui étaient peut-être trop ambitieux, et recentré nos efforts sur des problèmes qui semblaient plus abordables : les isomorphismes de graphes.

En effet, comme expliqué dans le chapitre 3, on peut constater l'existence d'un lien étroit entre les problèmes théoriques d'isomorphismes de graphes d'une part (isomorphisme de graphes et de sous-graphes, plus grand sous-graphe commun, voire distances entre graphes), et plusieurs applications liées à l'image d'autre part.

Aussi, nous avons dans le chapitre 4 développé une méthode d'extraction de graphes à partir d'images, avec pour intention première le respect de la sémantique. En effet, nous appuyant sur le fait que les graphes issus d'images sont plans, nous avons considéré

qu'il était raisonnable de s'attacher à donner un sens aux arêtes et aux faces, et de rendre la structure du graphe la plus significative possible. Pour cela, nous avons estimé que le plongement d'un graphe devait, visuellement, permettre une identification même partielle de l'image sous-jacente. La méthode explicitée dans ce chapitre répond de manière satisfaisante à ces contraintes, bien qu'elle reste dépendante de la segmentation et de l'extraction des points d'intérêt. Cependant, lorsque la segmentation respecte la sémantique, alors nous avons vu que le graphe obtenu était adéquat. Rappelons qu'il est également possible, étant donné le graphe, de revenir à l'image d'origine, avec une perte limitée.

Néanmoins, la représentation d'images par de *bons* graphes n'a pas constitué une finalité en soi, et le chapitre 5 a consisté à développer des algorithmes efficaces permettant de résoudre des problème intéressants à la fois parce qu'ils sont difficiles à mettre en œuvre sur les graphes, qui sont des objets théoriquement complexes, mais aussi pour leur intérêt vis-à-vis de l'image, qui reste notre domaine d'application privilégié.

Ainsi, les algorithmes présentés dans le dernier chapitre de cette thèse répondent aux problématiques d'isomorphismes de graphes et de sous-graphes plans exacts ; c'est-à-dire que l'on peut efficacement rechercher un motif dans un graphe, sans aucune variation.

Perspectives

Un outil pour l'extraction de graphes à partir d'images

Nous avons évoqué, dans la partie 4.3 du chapitre 4, l'existence d'une version de démonstration de notre méthode d'extraction de graphes à partir d'images, disponible en ligne[3]. Dans l'état actuel des choses, le cadre d'utilisation de cette démonstration est trop contraint : l'utilisateur n'est pas libre de choisir son algorithme de segmentation ou d'extraction de points d'intérêt, les graphes n'ont pas format de sortie défini, et on ne peut pas traiter les images par lots. Une perspective serait donc de refondre l'application, afin d'obtenir un programme pouvant être mis à disposition de la communauté, sous la forme d'un logiciel libre. De plus, il pourrait être intéressant d'enrichir les graphes de sortie par des labels (par exemple, des données numériques sur les sommets ou arêtes). En outre, puisque la triangulation a lieu par région de segmentation, chaque face du graphe appartient a une et une seule région, et pourrait alors être étiquetée par des données de la région.

Où est Charlie ?

Quant aux algorithmes du chapitre 5, ils s'avéreront très performants si nous les soumettons au jeu « Où est Charlie ? »[4]. Dans ce jeu, il s'agit principalement de retrouver le même personnage à l'identique, Charlie, dans une scène. La difficulté est liée au fait que les scènes sont très chargées et remplies de nombreux objets et de personnages fort ressemblants à Charlie. Aussi, si le graphe représentant notre ami est *correctement*

[3] http://labh-curien.univ-st-etienne.fr/EPG/
[4] http://findwally.co.uk/

extrait, s'il est assez caractéristique, alors Charlie sera rapidement trouvé, et nous ne tomberons pas dans les pièges des nombreux faux positifs possibles.

Ces travaux ont permis de construire des bases intéressantes et constituent un tremplin vers de nombreuses autres problématiques. En effet, outre les perspectives évoquées de la prise en compte des graphes non connexes, les applications visées peuvent être plus diversifiées : bien souvent, dans la vie et non dans les jeux, on ne souhaite pas retrouver Charlie, mais plutôt quelqu'un qui ressemble à Charlie (ce qui est bien tout le contraire du principe du jeu). C'est en ce point que nos travaux méritent d'être poursuivis. Effectivement, nous savons retrouver *la* tasse dans une image, et non pas *une* tasse dans une image. Pour des applications de la vie réelle, une perspective est donc de développer d'autres versions de ces algorithmes, qui nous permettraient de résoudre des isomorphismes tolérants. Il s'agirait par exemple de recherches à k faces, arêtes, sommets près.

Dans cette optique, ce travail constitue un premier pas vers le développement d'une mesure de distance entre graphes. En effet, deux graphes isomorphes à k faces près peuvent dès lors être considérés comme plus similaires que deux graphes isomorphes à $k+1$ faces près. Ainsi, il est envisageable de concevoir plusieurs critères permettant de mesurer la similarité entre graphes, critères pouvant être enrichis par l'introduction de labels (sur les faces, arêtes, sommets). Cette perspective n'est pas sans rappeler la distance d'édition entre graphes (étudiée dans la partie 3.3), qui a pour valeur la somme des coûts de suppressions, insertions et substitutions de faces, arêtes, sommets, permettant de passer d'un graphe à un autre.

En outre, on peut établir un parallèle entre la mise en place d'une mesure de distance entre graphes et le problème du plus grand sous-graphe commun (voir la partie 3.2.3), qui est une autre perspective de cette thèse. De ce point de vue, étant donné trois graphes, c'est par exemple la taille de leurs plus grands sous-graphes communs respectifs qui peut nous amener à détecter les couples de graphes les plus similaires.

Enfin, une autre piste à envisager concerne la recherche de motifs (sous-graphes) plans fréquents. En effet, les motifs (à condition que leur taille ne soit pas trop petite) portent en eux une partie de la sémantique des images dont ils sont issus. Dès lors, on peut penser que découvrir, dans des ensembles d'images, des groupes d'images ayant les mêmes motifs fréquents peut constituer un premier pas vers le développement de méthodes de classification.

Annexe

A

Diagramme de Voronoi et triangulation de Delaunay

Dans cette annexe, nous nous proposons de détailler les principes régissant la construction d'un diagramme de Voronoi et de son dual, la triangulation de Delaunay.

A.1 Diagramme de Voronoi

Comme nous le proposent [de Berg et al., 2000], imaginons que nous gérons une chaîne de magasins : nous possédons ainsi la carte de leurs emplacements géographiques respectifs. Pour estimer le nombre de clients que chacun peut attirer, on peut supposer que tout client se rendra dans le magasin le plus proche de son domicile (voir la figure A.i[1]). De même, si nous souhaitons construire un nouveau magasin, ce que nous appellerons les zones d'influences de ceux déjà construits, mais aussi de ceux de nos concurrents, sont à prendre en compte. Le diagramme de Voronoi répond à ce genre de problématiques.

Soyons plus formels. Soit P un ensemble de n points, que l'on appellera sites, dans un espace euclidien E^d de dimension d (par la suite, nous posons $d = 2$). La problématique à laquelle se propose de répondre la structure géométrique que constitue le diagramme de Voronoi est liée à celle de recherche du plus proche voisin. Étant donné un point quelconque de l'espace, quel est le site qui en est le plus proche ? Ce modèle, où l'on assigne à chaque point le site le plus proche, est appelé *modèle d'assignation de Voronoi*. La subdivision induite par ces assignations est appelée *diagramme de Voronoi* [Dirichlet, 1850, Voronoi, 1907, Voronoi, 1908].

Définition A.1 (Diagramme de Voronoi)
Soit P un ensemble de n sites et soit $d(p_i, p_j)$ une distance entre deux sites p_i et p_j (ordinairement, il s'agit de la distance euclidienne). Le diagramme de Voronoi se définit comme la subdivision de l'espace en n cellules, une pour chaque site de P, avec la propriété suivante : un point q appartient à une cellule correspondant à un site p_i si

[1] La plupart des diagrammes de Voronoi et triangulations de Delaunay illustrant ce chapitre ont été réalisées avec l'applet Voroglide disponible en ligne http://www.pi6.fernuni-hagen.de/GeomLab/VoroGlide/index.html.en

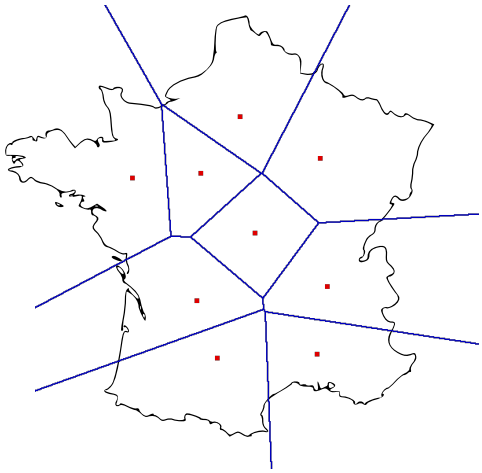


FIG. A.i – Zones d'influences de magasins (points rouges) dénotées par le diagramme de Voronoi.

et seulement si $d(q,p_i) < d(q,p_j)$ *pour tout* $j \in P$*, avec* $i \neq j$*. On note* $Vor(P)$ *le diagramme de Voronoi de* P*.*

$Vor(P)$ désigne les sommets et les arêtes de la subdivision. La cellule correspondant à un site p_i est notée $\mathcal{V}(p_i)$.

Soit la médiatrice de deux sites p_i et p_j. Celle-ci découpe l'espace en deux demi-plans, notés $h(p_i,p_j)$ (demi-plan contenant p_i) et $h(p_j,p_i)$ (demi-plan contenant p_j). Ainsi, un point $q \in h(p_i,p_j)$ si et seulement si $d(q,p_i) < d(q,p_j)$. On obtient donc :

$$\mathcal{V}(p_i) = \cap_{1\leq j\leq n,\ j\neq i}\ h(p_i,p_j)$$

$\mathcal{V}(p_i)$ est ainsi l'intersection de $n-1$ demi-plans, et constitue un polygone (borné ou non) convexe, d'au plus $n-1$ sommets et $n-1$ arêtes.

Le théorème suivant est prouvé dans [de Berg et al., 2000] :

Théorème A.1 *Soit* P *un ensemble de* n *sites. Si tous sont colinéaires, alors* $Vor(P)$ *est formé de* $n-1$ *lignes parallèles. Sinon,* $Vor(P)$ *est connexe et ses arêtes sont des segments ou des demi-droites.*

La figure A.ii représente un diagramme de Voronoi d'un ensemble de sites en dimension 2. Les cellules non bornées correspondent aux régions des sites de l'enveloppe convexe de P, qui est représentée en vert.

Définition A.2 (Enveloppe convexe)
L'enveloppe convexe d'un ensemble de points P *est le plus petit polygone convexe contenant* P*. Un polygone est convexe si tout segment joignant deux des ses points est entièrement inclus dans le polygone.*

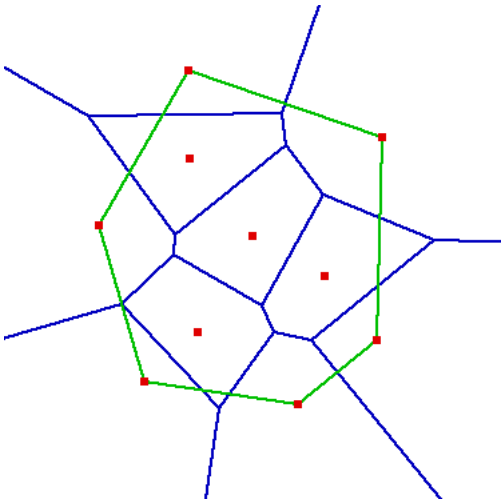


FIG. A.ii – Diagramme de Voronoi d'un ensemble de sites en dimension 2 et enveloppe convexe associée (en vert).

Outre des sites, un diagramme de Voronoi contient donc également des arêtes, et des sommets (qui se trouvent aux intersections des extrémités des arêtes). Sommets et arêtes peuvent être caractérisés comme suit :

Théorème A.2 *Soit P un ensemble de sites et $C_P(q)$ le plus grand cercle de centre q (point de l'espace) ne contenant aucun site de P.*

i q est un sommet de $Vor(P)$ si et seulement si $C_P(q)$ contient au moins 3 sites sur sa frontière ;

ii la médiatrice de deux sites p_i et p_j est une arête de $Vor(P)$ si et seulement si il existe un point r sur cette médiatrice, tel que $C_P(r)$ contient p_i et p_j sur sa frontière, à l'exception de tout autre site.

Toute médiatrice des sites de n ne fait donc pas forcément partie du diagramme de Voronoi correspondant. On a les informations suivantes :

Théorème A.3 *Pour $n \geq 3$, le nombre de sommets d'un diagramme de Voronoi d'un ensemble de n sites est au plus de $2n - 5$, et le nombre d'arêtes est au plus de $3n - 6$.*

Le diagramme de Voronoi d'un ensemble de site peut se calculer, par l'algorithme *plane sweep* de Fortune, en $\mathcal{O}(n \log n)$ [Fortune, 1992].

Notons qu'il existe des variantes du diagramme de Voronoi : certaines proposent de changer la fonction de distance utilisée entre les sites, ou d'associer des poids à ces derniers. D'autres encore proposent de partitionner l'espace en fonction des k plus proches voisins, ou en fonction du site le plus éloigné.

A.2 Triangulation

Le diagramme de Voronoi possède une structure duale, la triangulation de Delaunay.

A.2.1 Définitions et propriétés

Soit P un ensemble de n points appelés sites, dans un espace euclidien E^2. On définit une triangulation $\mathcal{T}$ de P comme suit :

Définition A.3 (Triangulation)
Une triangulation $\mathcal{T}$ de P est une subdivision planaire dont les faces bornées sont des triangles, et dont les sommets sont les sites de P.

Une triangulation $\mathcal{T}$ est ainsi constituée de faces bornées et d'une face non bornée (face extérieure ou infinie), qui est le complément de l'enveloppe convexe de P. Chaque face bornée est un triangle.

La notion de triangulation peut également être liée à celle de subdivision planaire maximale : il s'agit d'une subdivision planaire maximale dont les sommets sont les points de P [de Berg et al., 2000] :

Définition A.4 (Subdivision planaire maximale)
Une subdivision planaire maximale est une subdivision S telle que tout ajout d'arête à S détruise sa planarité.

De plus, toute arête de l'enveloppe convexe de P est une arête de $\mathcal{T}$, et ce quelle que soit la triangulation. En effet, il existe généralement plusieurs triangulations possibles d'un même ensemble de points. Certaines possèdent des propriétés particulières, qui les font être préférables à d'autres selon les domaines d'application. C'est le cas de la triangulation de Delaunay, que nous étudierons dans la partie A.2.3. Cependant, quelle que soit la triangulation de P, les nombres de triangles et d'arêtes, eux, ne changent pas. Ils varient uniquement en fonction du nombre de points de P qui sont sur l'enveloppe convexe :

Théorème A.4 *Soit P un ensemble de n points n'étant pas tous colinéaires, et soit k le nombre de points de P étant sur son enveloppe convexe. Alors, toute triangulation de P a $2n-2-k$ triangles et $3n-3-k$ arêtes.*

Supposons que $\mathcal{T}$, une triangulation de P, ait t triangles. On peut ainsi ordonner les $3t$ angles résultants dans un vecteur d'ordre croissant $A(\mathcal{T}) = (\alpha_1,\ \alpha_2,\ \ldots,\ \alpha_{3t})$, où, $\forall i < j,\ \alpha_i \leq \alpha_j$. Soit $\mathcal{T}'$ une autre triangulation de P, avec $A(\mathcal{T}') = (\alpha'_1,\ \alpha'_2,\ \ldots,\ \alpha'_{3t})$. On dit que $A(\mathcal{T})$ est plus grand que $A(\mathcal{T}')$ si $A(\mathcal{T})$ est lexicographiquement plus grand que $A(\mathcal{T}')$, c'est-à-dire qu'il existe i, avec $1 \leq i \leq 3t$, tel que :

$$\alpha_j = \alpha'_j \quad \forall j < i, \quad et \quad \alpha_i > \alpha'_i.$$

Ceci se note $A(\mathcal{T}) > A(\mathcal{T}')$. Une triangulation $A(\mathcal{T})$ est *angle-optimale* si $A(\mathcal{T}) \geq A(\mathcal{T}')$ pour toute triangulation $A(\mathcal{T}')$ de P.

Il existe parfois, lors de la construction d'une triangulation, plusieurs possibilités pour placer une arête entre deux sommets. Ce cas est illustré sur la figure A.iii.

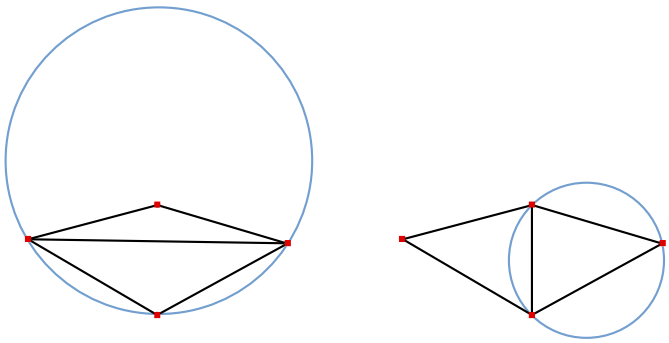


FIG. A.iii – Deux triangulation possibles d'un même quadrilatère. Celle de gauche contient une arête illégale.

La triangulation $\mathcal{T}$ de gauche fait apparaître une arête dite illégale, ce qui est le cas si l'on peut localement augmenter le plus petit angle en intervertissant l'arête en question. Ainsi, la triangulation $\mathcal{T}'$ de droite ne contient pas d'arête illégale et $A(\mathcal{T}') > A(\mathcal{T})$. Le lemme suivant précise le cas où une arête est illégale [de Berg et al., 2000] :

Lemme A.1 (Arête illégale) *Soit une arête p_ip_j incidente aux triangles $p_ip_jp_k$ et $p_ip_jp_l$, et soit C le cercle circonscrit aux points p_i, p_j et p_k. L'arête p_ip_j est illégale si et seulement si p_l est à l'intérieur de C. De plus, si les points p_i, p_j, p_k et p_l forment un quadrilatère convexe et ne sont pas cocirculaires, alors exactement une des deux arêtes p_ip_j et p_kp_l est illégale.*

On définit alors une triangulation *légale* comme ne contenant aucune arête illégale. Il s'en suit que toute triangulation angle-optimale est légale.

A.2.2 Graphe de Delaunay

Nous avons étudié dans la partie A.1 le diagramme de Voronoi $Vor(P)$ d'un ensemble P de n sites. Ce diagramme subdivise l'espace en n régions, une pour chaque site de P, de façon à ce que la région $\mathcal{V}(p)$ d'un site p (aussi appelée cellule) corresponde à l'ensemble des points plus proches de p que de tout autre site.

Définition A.5 (Graphe de Delaunay)
Soit $\mathcal{G}$ le dual du diagramme de Voronoi : à chaque site de P correspond un sommet de $\mathcal{G}$, et il existe une arête entre deux sommets si leurs cellules de Voronoi ont une frontière commune. Ceci signifie également qu'à toute arête de $Vor(P)$ correspond une arête dans $\mathcal{G}$. Alors, $\mathcal{G}$ est appelé le graphe de Delaunay de P, et noté $\mathcal{DG}(P)$.

D'après le théorème A.3, la dualité de ces deux structures engendre le fait que le graphe de Delaunay possède au plus $3n - 6$ arêtes.

Le théorème suivant est prouvé dans [de Berg et al., 2000] :

Théorème A.5 *Le graphe de Delaunay d'un ensemble de points planaires est un graphe planaire.*

Le graphe de Delaunay est constitué de polygones convexes aux nombres de sommets variables. Un exemple est illustré sur la figure A.iv : si plus de 3 points sont cocirculaires, alors le graphe n'est pas formé uniquement de triangles. Sur l'exemple présent, cinq points sont cocirculaires et forment donc un pentagone. Ceci signifie que le diagramme de Voronoi correspondant a un sommet de degré cinq.

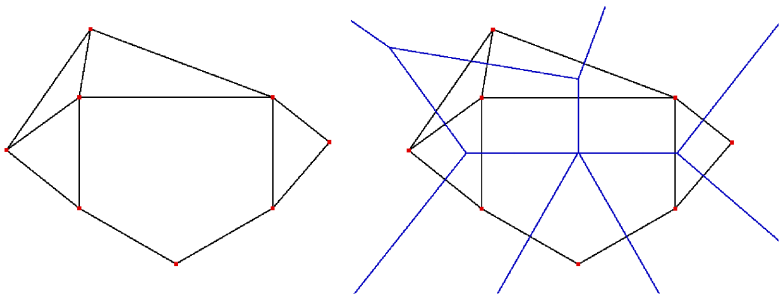


FIG. A.iv – Graphe de Delaunay d'un ensemble de points en dimension 2.

A.2.3 Triangulation de Delaunay

L'éventualité d'un ensemble de sites où plus de trois d'entre eux sont cocirculaires constitue un cas dit dégénéré. Dans la situation contraire, on dit que les sites sont en *position générale*. Le diagramme de Voronoi de sites en position générale ne contient que des sommets de degré 3, et le graphe de Delaunay dual est formé uniquement de triangles. On parle alors de *triangulation de Delaunay*.

La figure A.v présente la triangulation de Delaunay de l'ensemble de points correspondant au diagramme de Voronoi de la figure A.ii.

Définition A.6 (Triangulation de Delaunay)
On définit la triangulation de Delaunay comme étant toute triangulation obtenue en ajoutant des arêtes au graphe de Delaunay. La triangulation de Delaunay est unique si et seulement si P est en position générale.

La figure A.vi représente ainsi une triangulation du graphe de Delaunay de la figure A.iv.

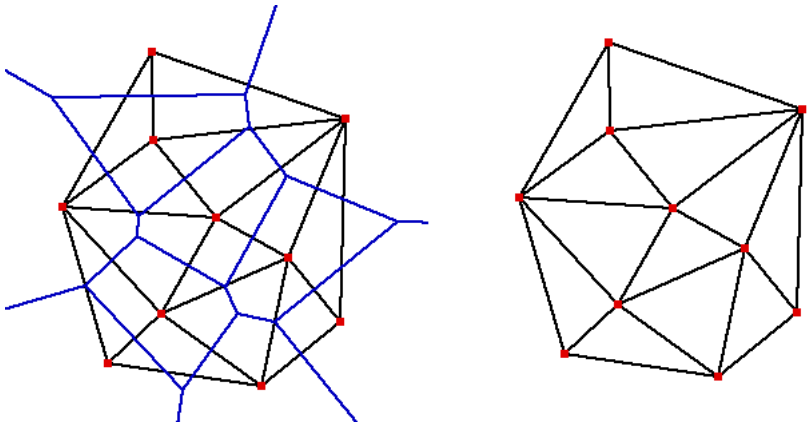


FIG. A.v – Triangulation de Delaunay correspondant au diagramme de Voronoi A.ii.

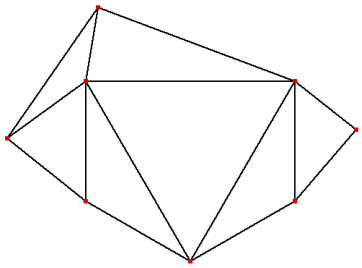


FIG. A.vi – Triangulation du graphe de Delaunay A.iv.

Plusieurs théorèmes sont associés à la triangulation de Delaunay [de Berg et al., 2000, Hjelle et Dæhlen, 2006] :

Théorème A.6 *Soit P un ensemble de sites.*

i trois points p_i, p_j et $p_k \in P$ sont les sommets d'une face de la triangulation de Delaunay si et seulement si le cercle circonscrit aux sommets ne contient aucun autre site ;

ii deux points p_i et $p_j \in P$ sont reliés par une arête dans la triangulation de Delaunay si et seulement si il existe un cercle ayant p_i et p_j sur sa frontière, et ne contenant aucun autre site.

Le centre du cercle circonscrit à une face de le triangulation de Delaunay correspond à un sommet du diagramme de Voronoi dual.

Théorème A.7 *Soit P un ensemble de sites, et $\mathcal{T}$ une triangulation de P. Alors $\mathcal{T}$ est une triangulation de Delaunay de P si et seulement si le cercle circonscrit à tout triangle de $\mathcal{T}$ ne contient aucun point de P.*

Théorème A.8 *Soit P un ensemble de sites. Une triangulation $\mathcal{T}$ de P est légale (ne contient aucune arête illégale) si et seulement si $\mathcal{T}$ est une triangulation de Delaunay de P.*

Ce dernier théorème implique que toute triangulation angle-optimale de P est une triangulation de Delaunay de P. De plus, toute triangulation de Delaunay de P maximise le plus petit angle vis-à-vis de toute triangulation possible de P. Cette caractéristique rend préférable l'utilisation de la triangulation de Delaunay dans certains domaines, notamment en reconstruction d'image. Il s'agit alors de représenter une image par une triangulation de Delaunay ayant pour sommets les pixels les plus importants de celle-ci. En partant de la triangulation, il est ensuite possible de reconstruire l'image de départ (avec perte) en interpolant les valeurs des pixels à l'intérieur des triangles. L'interpolation est rendue plus aisée et le résultat est généralement plus probant lorsque les angles ne sont pas trop aigus [Barnhill, 1977].

La triangulation de Delaunay est également particulière vis-à-vis des rayons des plus petits cercles contenant les faces de celle-ci. Si un triangle est aigu (ses trois angles sont inférieurs à 90°), alors le plus petit cercle le contenant est son cercle circonscrit. Par contre, s'il est obtus (un de ses angles est supérieur à 90°), alors le plus petit cercle le contenant a son centre au milieu de l'arête la plus longue. Pour les triangles rectangle, le plu petit cercle vérifie les deux propriétés. Appelons ce cercle le plus petit cercle contenant. Soit $maxray(\mathcal{T})$ le plus grand rayon des plus petits cercles contenant de la triangulation $\mathcal{T}$. Alors, la triangulation de Delaunay minimise $maxray(\mathcal{T})$ [Fortune, 1992].

Enfin, comme nous l'avons évoqué dans la partie 1.3.2, la triangulation de Delaunay peut être mise en relation avec certains graphes remarquables, que sont :

- l'arbre couvrant de longueur minimale $\mathcal{EMST}$ (*Euclidean minimum spanning tree*) qui minimise la somme des longueurs euclidiennes des arêtes de l'arbre ;
- le graphe des voisins relatifs $\mathcal{RNG}$ (*relative neighbourhood graph*) pour lequel il existe une arête entre deux sommets s_1 et s_2 si l'intersection des cercles de centre s_1 et s_2 et de rayon s_1s_2 est vide ;
- le graphe de Gabriel $\mathcal{GG}$ pour lequel il existe une arête entre deux sommets s_1 et s_2 si le cercle de diamètre s_1s_2 est vide.

Les arêtes de ces graphes vérifient [Fortune, 1992] :

$$\mathcal{EMST} \subseteq \mathcal{RNG} \subseteq \mathcal{GG} \subseteq \mathcal{T}$$

La construction du diagramme de Voronoi d'un ensemble de n sites requiert une complexité de calcul de $\mathcal{O}(n \log n)$ et de stockage de $\mathcal{O}(n)$. Il est possible, à partir du diagramme de Voronoi, d'en déduire la triangulation de Delaunay duale, en $\mathcal{O}(n)$. En outre, il existe des algorithmes, comme le *randomized incremental* [Guibas et al., 1990, Guibas et al., 1992], qui permettent le calcul direct de la Triangulation de Delaunay de n sites avec une complexité de calcul de $\mathcal{O}(n \log n)$ et de stockage de $\mathcal{O}(n)$.

Bibliographie

[Alliez et al., 2008] Alliez, P., Attene, M., Gotsman, C., et Ucelli, G. (2008). Recent advances in remeshing of surfaces. *Shape Analysis and Structuring*, pages 53–82. *(Cité page 32)*

[Ambauen et al., 2003] Ambauen, R., Fischer, S., et Bunke, H. (2003). Graph edit distance with node splitting and merging, and its application to diatom identification. Dans *Proceedings of the 4th IAPR Workshop on Graph-based Representations in Pattern Recognition (GbRPR 03)*, volume 2726 de *Lecture Notes in Computer Science*, pages 259–264. Springer. *(Cité page 83)*

[Applegate et al., 2006] Applegate, D. L., Bixby, R. E., Chvátal, V., et Cook, W. J., editors (2006). *The traveling salesman problem : a computational study*. Princeton University Press. *(Cité page 7)*

[Baeza-Yates, 1998] Baeza-Yates, R. A. (1998). Similarity in two-dimensional strings. Dans *Proceedings of the 14th International Computing and Combinatorics Conference (COCOON 98)*, pages 319–328. Springer Berlin/Heidelberg. *(Cité page 63)*

[Barnard et al., 2003] Barnard, K., Duygulu, P., Forsyth, D. A., de Freitas, N., Blei, D. M., et Jordan, M. I. (2003). Matching words and pictures. *Journal of Machine Learning Research*, 3 : 1107–1135. *(Cité page 9)*

[Barnhill, 1977] Barnhill, R. E. (1977). Representation and approximation of surfaces. Dans Rice, J. R., editor, *Mathematical Software III*, pages 69–120. Academic Press. *(Cité page 172)*

[Bentley, 1975] Bentley, J. L. (1975). Multidimensional binary search trees used for associative searching. *Communications of the ACM*, 18(9) : 509–517. *(Cité page 27)*

[Bernard et al., 2008] Bernard, M., Boyer, L., Habrard, A., et Sebban, M. (2008). Learning probabilistic models of tree edit distance. *Pattern Recognition*, 41 : 2611–2629. *(Cité page 27)*

[Berretti et al., 2004] Berretti, S., Bimbo, A. D., et Pala, P. (2004). A graph edit distance based on node merging. Dans *Proceedings of Conference on Image and Video Retrieval (CIVR 04)*, pages 464–472. *(Cité page 83)*

[Bhowmick et Bhattacharya, 2007] Bhowmick, P. et Bhattacharya, B. (2007). Fast polygonal approximation of digital curves using relaxed straightness properties. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(9) : 1590–1602. *(Cité page 91)*

[Bille, 2005] Bille, P. (2005). A survey on tree edit distance and related problems. *Theoretical Computer Science*, 337(1-3) : 217–239. *(Cité page 27)*

[Billingsley, 1965] Billingsley, F. C. (1965). Digital video processing at JPL. Dans *Proceedings of Electronic Imaging Techniques I (SPIE 65)*, pages XV–1–19, Bellingham, WA : Society for Photo-Optical Instrumentation Engineers. Eugene B. Turner. *(Cité page 14)*

[Blum, 1967] Blum, H. (1967). A transformation for extracting descriptors of shape. Dans *Models for the Perception of Speech and Visual Forms*, pages 362–380. MIT Press. *(Cité page 23)*

[Bolon et al., 1995] Bolon, P., Chassery, J.-M., Cocquerez, J.-P., Demigny, D., Graffigne, C., Montanvert, A., Philipp, S., Zéboudj, R., et Zérubia, J., editors (1995). *Analyse d'images : filtrage et segmentation*. Masson. *(Cité page 20)*

[Bonchev et Rouvray, 1991] Bonchev, D. et Rouvray, D. H., editors (1991). *Chemical graph theory : introduction and fundamentals*. Taylor & Francis. *(Cité page 7)*

[Bres et Jolion, 1999] Bres, S. et Jolion, J.-M. (1999). Detection of interest points for image indexation. Dans *Proceedings of the 3rd International Conference on Visual Information and Information Systems (VISUAL 99)*, volume 1614 de *Lecture Notes in Computer Science*, pages 427–434. Springer Berlin/Heidelberg. *(Cité pages 40, 53 et 102)*

[Bunke, 1997] Bunke, H. (1997). On a relation between graph edit distance and maximum common subgraph. *Pattern Recognition Letters*, 18 : 689–694. *(Cité page 82)*

[Bunke, 1998] Bunke, H. (1998). Error-tolerant graph matching : a formal framework and algorithms. Dans *Proceedings of the 7th IAPR International Workshop on Structural and Syntactic Pattern Recognition (SSPR 98)*, pages 1–14. Springer-Verlag. *(Cité page 81)*

[Bunke, 2000] Bunke, H. (2000). Graph matching : theoretical foundations, algorithms, and applications. Dans *International Conference on Vision Interface*, pages 82–84. *(Cité page 69)*

[Bunke et Csirik, 1995] Bunke, H. et Csirik, J. (1995). Parametric string edit distance and its application to pattern recognition. Dans *IEEE Transactions on Systems, Man and Cybernetics*, volume 25. *(Cité page 25)*

[Bunke et al., 2002] Bunke, H., Jiang, X., Abegglen, K., et Kandel, A. (2002). On the weighted mean of a pair of strings. *Pattern analysis and applications*, 5(1) : 23–30. *(Cité page 25)*

[Bunke et Shearer, 1998] Bunke, H. et Shearer, K. (1998). A graph distance metric based on the maximal common subgraph. *Pattern Recognition Letters*, 19 : 255–259. *(Cité page 81)*

[Burt et Adelson, 1983] Burt, P. J. et Adelson, E. H. (1983). The laplacian pyramid as a compact image code. *IEEE Transactions on Communications*, 31 : 532–540. *(Cité page 16)*

[Caelli et Kosinov, 2004] Caelli, T. et Kosinov, S. (2004). An eigenspace projection clustering method for inexact graph matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26 : 515–519. *(Cité page 78)*

[Chapelle et al., 1999] Chapelle, O., Haffner, P., et Vapnik, V. (1999). Support vector machines for histogram-based image classification. *IEEE Transactions on Neural Networks*, 10(5) : 1055–1064. *(Cité page 24)*

[Chen, 1998] Chen, W. (1998). More efficient algorithm for ordered tree inclusion. *Journal of Algorithms*, 26(2) : 370–385. *(Cité page 28)*

[Choset, 2005] Choset, H. M., editor (2005). *Principles of robot motion : theory, algorithms, and implementation*. MIT Press. *(Cité page 113)*

[Coeurjolly et al., 2007] Coeurjolly, D., Montanvert, A., et Chassery, J.-M., editors (2007). *Géométrie discrète et images numériques*. Traité IC2, Signal et Image. Hermès Paris, France. *(Cité pages 14 et 15)*

[Conte et al., 2004] Conte, D., Foggia, P., Sansone, C., et Vento, M. (2004). Thirty years of graph matching in pattern recognition. *International Journal of Pattern Recognition and Artificial Intelligence*. *(Cité page 69)*

[Conte et al., 2007] Conte, D., Foggia, P., Sansone, C., et Vento, M. (2007). How and why pattern recognition and computer vision applications use graphs. Dans *Applied Graph Theory in Computer Vision and Pattern Recognition*, pages 85–135. Springer. *(Cité pages 7 et 69)*

[Cordella et al., 1999] Cordella, L. P., Foggia, P., Sansone, C., et Vento, M. (1999). Performance evaluation of the vf graph matching algorithm. Dans *Proceedings of the 10th International Conference on Image Analysis and Processing (ICIAP 99)*, pages 1172–1177. IEEE Computer Society. *(Cité page 77)*

[Cordella et al., 2001] Cordella, L. P., Foggia, P., Sansone, C., et Vento, M. (2001). An improved algorithm for matching large graphs. Dans *Proceedings of the 3rd IAPR Workshop on Graph-based Representations in Pattern Recognition (GbRPR 01)*, pages 149–159. *(Cité pages 77 et 152)*

[Cori, 1973] Cori, R. (1973). *Un code pour les graphes planaires et ses applications*. Thèse de doctorat, Université Paris VII. *(Cité page 153)*

[Cori, 1975] Cori, R. (1975). Un code pour les graphes planaires et ses applications. Dans *Astérisque*, volume 27. Société Mathématique de France, Paris, France. *(Cité pages 30 et 153)*

[Cori, 1985] Cori, R. (1985). Computation of the automorphism group of a topological graph embedding. Technical Report I-8612, UER de mathématique et informatique, CNRS équipe du laboratoire associé 226, Université Bordeaux I. *(Cité page 155)*

[Cormen et al., 1990] Cormen, T. H., Leiserson, C. E., Rivest, R. L., et Stein, C. (1990). *Introduction to algorithms*. MIT Press and McGraw-Hill, 1rst edition. *(Cité pages 8 et 43)*

[Culik II et Kari, 1997] Culik II, K. et Kari, J. (1997). Digital images and formal languages. Dans *Handbook of Formal Languages*, volume 3 : Beyond Words, pages 599–616. Springer-Verlag New York, Inc. *(Cité page 23)*

[Damiand et al., 2004] Damiand, G., Bertrand, Y., et Fiorio, C. (2004). Topological model for two-dimensional image representation : definition and optimal extraction algorithm. *Computer Vision and Image Understanding*, 93(2) : 111–154. *(Cité page 150)*

[Damiand et Coeurjolly, 2008] Damiand, G. et Coeurjolly, D. (2008). A generic and parallel algorithm for 2d image discrete contour reconstruction. Dans *Proceedings of the 4th International Symposium on Visual Computing (ISVC 08)*, volume 5359 de *Lecture Notes in Computer Science*, pages 792–801. Springer Berlin/Heidelberg. *(Cité page 91)*

[Damiand et al., 2009a] Damiand, G., de la Higuera, C., Janodet, J.-C., Samuel, É., et Solnon, C. (2009a). A polynomial algorithm for subisomorphism of open plane graphs. Dans *Proceedings of the 7th International Workshop on Mining and Learning with Graphs (MLG 09)*, page poster. *(Cité page 12)*

[Damiand et al., 2009b] Damiand, G., de la Higuera, C., Janodet, J.-C., Samuel, É., et Solnon, C. (2009b). A polynomial algorithm for submap isomorphism : application to searching patterns in images. Dans *Proceedings of the 7th IAPR International Workshop on Graph-based Representations in Pattern Recognition (GbRPR 09)*, volume 5534 de *Lecture Notes in Computer Science*, pages 102–112. *(Cité pages 12 et 153)*

[Damiand et al., 2011] Damiand, G., Solnon, C., de la Higuera, C., Janodet, J.-C., et Samuel, É. (2011). Polynomial algorithms for subisomorphism of nd open combinatorial maps. *Computer Vision and Image Understanding*, 115(7) : 996–1010. *(Cité pages 12, 153 et 155)*

[de Berg et al., 2000] de Berg, M., van Kreveld, M., Overmars, M., et Schwarzkopf, O. (2000). *Computational geometry - algorithms and applications*. Springer Berlin/Heidelberg, 2nd edition. *(Cité pages 165, 166, 168, 169, 170 et 171)*

[De Santo et al., 2003] De Santo, M., Foggia, P., Sansone, C., et Vento, M. (2003). A large database of graphs and its use for benchmarking graph isomorphism algorithms. *Pattern Recognition Letters*, 24 : 1067–1079. *(Cité page 7)*

[Deng et al., 2006] Deng, H., Mortensen, E., Shapiro, L., et Dietterich, T. (2006). Reinforcement matching using region context. Dans *Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 06)*, page 11. *(Cité page 68)*

[Dent et al., 2008] Dent, B., Torguson, J., et Hodler, T., editors (2008). *Cartography : thematic map design*. McGraw-Hill Science/Engineering/Math. *(Cité page 7)*

[Dirichlet, 1850] Dirichlet, G. L. (1850). Über die reduktion der positiven quadratischen formen mit drei unbestimmten ganzen zahlen. *Journal für die Reine und Angewandte Mathematik*, 40 : 209–227. *(Cité page 165)*

[Eco, 1970] Eco, U. (1970). Sémiologie des messages visuels. *Communications*, 15 : 11–51. *(Cité page 5)*

[Edmonds, 1960] Edmonds, J. (1960). A combinatorial representation for polyhedral surfaces. *Notices of the American Mathematical Society*, 7. *(Cité page 30)*

[Emile Aarts, 2003] Emile Aarts, J. K. L., editor (2003). *Local Search in combinatorial optimization*. Princeton University Press. *(Cité page 38)*

[Eppstein, 1995] Eppstein, D. (1995). Subgraph isomorphism in planar graphs and related problems. Dans *Proceedings of the 6th annual ACM-SIAM Symposium On Discrete Algorithms (SODA 95)*, pages 632–640. Society for Industrial and Applied Mathematics. *(Cité page 80)*

[Fáry, 1948] Fáry, I. (1948). On straight-line representation of planar graphs. *Acta Scientiarum Mathematicarum*, 11 : 229–233. *(Cité pages 73 et 120)*

[Finkel et Bentley, 1974] Finkel, R. A. et Bentley, J. L. (1974). Quad trees : a data structure for retrieval on composite keys. *Acta Informatica*, 4 : 1–9. *(Cité page 26)*

[Fiorio, 1995] Fiorio, C. (1995). *Approche interpixel en analyse d'images : une topologie et des algorithmes de segmentation.* Thèse de doctorat, Université Montpellier II. *(Cité page 87)*

[Fortin, 1996] Fortin, S. (1996). The graph isomorphism problem. Technical report, Department of Computing Science, University of Alberta, Edmonton, Canada. *(Cité pages 76 et 77)*

[Fortune, 1992] Fortune, S. (1992). Voronoi diagrams and delaunay triangulations. Dans Du, D.-Z. et Hwang, F., editors, *Computing in Euclidean Geometry*, Lecture Notes Series on Computing, pages 193–233. World Scientific. *(Cité pages 32, 167 et 172)*

[Françon, 1996] Françon, J. (1996). On recent trends in discrete geometry in computer science. Dans *Proceedings of the 6th International Conference on Discrete Geometry for Computer Imagery (DGCI 96)*, pages 3–16. *(Cité page 87)*

[Franz et al., 2005] Franz, G., Mallot, H. A., et Wiener, J. M. (2005). Graph-based models of space in architecture and cognitive science - a comparative analysis. *Architecture, Engineering and Construction of Build Environments*, pages 30–38. *(Cité page 7)*

[Freeman, 1961] Freeman, H. (1961). On the encoding of arbitrary geometric configurations. *IEEE Transactions on Computers*, 10(2) : 260–268. *(Cité page 24)*

[Fusy, 2007] Fusy, É. (2007). *Combinatoire des cartes planaires et applications algorithmiques.* Thèse de doctorat, LIX, École Polytechnique. *(Cité pages 73 et 114)*

[Gabriel et Sokal, 1969] Gabriel, K. R. et Sokal, R. R. (1969). A new statistical approach to geographic variation analysis. *Systematic Zoology*, 18 : 259–278. *(Cité page 32)*

[Garcia et al., 1999] Garcia, M., Sappa, A., et Vintimilla, B. (1999). Efficient approximation of gray-scale images through bounded error triangular meshes. Dans *Proceedings of the 1999 International Conference on Image Processing (ICIP 99)*, pages 168–170. *(Cité page 32)*

[Garey et Johnson, 1979] Garey, M. R. et Johnson, D. S. (1979). *Computers and intractability : a guide to the theory of NP-completeness.* W. H. Freeman & Co., New York, NY, USA. *(Cité page 79)*

[Gold et Rangarajan, 1996] Gold, S. et Rangarajan, A. (1996). A graduated assignment algorithm for graph matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(4) : 377–388. *(Cité pages 57 et 78)*

[Gonzalez et Woods, 2006] Gonzalez, R. C. et Woods, R. E. (2006). *Digital image processing.* Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 3rd edition. *(Cité pages 19 et 20)*

[Guibas et al., 1990] Guibas, L. J., Knuth, D. E., et Sharir, M. (1990). Randomized incremental construction of delaunay and voronoi diagrams. Dans *Proceedings of the 17th International Colloquium on Automata, Languages and Programming*

(ICALP 90), pages 414–431, New York, NY, USA. Springer-Verlag New York, Inc. *(Cité page 172)*

[Guibas et al., 1992] Guibas, L. J., Knuth, D. E., et Sharir, M. (1992). Randomized incremental construction of delaunay and voronoi diagrams. *Algorithmica*, 7(4) : 381–413. *(Cité page 172)*

[Hamming, 1950] Hamming, R. W. (1950). Error detecting and error correcting codes. *Bell System Technical Journal*, 29(2) : 147–160. *(Cité page 41)*

[Harris et Stephens, 1988] Harris, C. et Stephens, M. (1988). A combined corner and edge detection. Dans *Proceedings of The Fourth Alvey Vision Conference*, pages 147–151. *(Cité page 21)*

[Hart et al., 1968] Hart, P., Nilsson, N., et Raphael, B. (1968). A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics*, 4 : 100 – 107. *(Cité page 83)*

[Hjelle et Dæhlen, 2006] Hjelle, Ø. et Dæhlen, M. (2006). *Triangulations and applications (mathematics and visualization)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA. *(Cité page 171)*

[Hong et al., 2004] Hong, S.-H., Merrick, D., et do Nascimento, H. A. D. (2004). The metro map layout problem. Dans *Proceedings of the 2004 Australasian symposium on Information Visualisation (APVis 0)*, pages 91–100. Australian Computer Society, Inc. *(Cité page 7)*

[Hopcroft et Wong, 1974] Hopcroft, J. E. et Wong, J. K. (1974). Linear time algorithm for isomorphism of planar graphs. Dans *Proceedings of the 6th annual ACM Symposium on Theory Of Computing (STOC 74)*, pages 172–184. ACM. *(Cité page 78)*

[Horowitz et Pavlidis, 1976] Horowitz, S. L. et Pavlidis, T. (1976). Picture segmentation by a tree traversal algorithm. *Journal of the ACM*, 23(2) : 368–388. *(Cité page 20)*

[Jaja et Kosaraju, 1988] Jaja, J. et Kosaraju, S. (1988). Parallel algorithms for planar graph isomorphism and related problems. *IEEE Transactions on Circuits and Systems*, 35(3) : 304 – 311. *(Cité page 78)*

[Jiang et al., 1994] Jiang, T., Wang, L., et Zhang, K. (1994). Alignment of trees - an alternative to tree edit. Dans *Proceedings of the 5th Annual Symposium on Combinatorial Pattern Matching (CPM 94)*, pages 75–86. Springer Berlin/Heidelberg. *(Cité page 28)*

[Jiang et al., 2004] Jiang, X., Bunke, H., et Csirik, J. (2004). Median strings : a review. *Data Mining in Time Series Databases*, 57 : 173–192. *(Cité page 25)*

[Jiten et Mérialdo, 2007] Jiten, J. et Mérialdo, B. (2007). Semantic image segmentation with a multidimensional hidden markov model. Dans *Multimedia Modeling MMM*, pages 616–624. *(Cité page 87)*

[Jolion et Simand, 2004] Jolion, J. M. et Simand, I. (2004). Représentation d'images par des chaînes de symboles : application à l'indexation d'images. Dans *Proceedings of the 2004 Conférence sur la COmpression et REprésentation des Signaux Audiovisuels (CORESA 04)*. *(Cité page 42)*

[Kari, 2006] Kari, J. (2006). Image processing using finite automata. Dans *Recent Advances in Formal Languages and Applications*, pages 171–208. Springer Berlin/Heidelberg. *(Cité page 23)*

[Kendall, 1970] Kendall, M. G. (1970). *Rank correlation methods.* Charles Griffin. *(Cité page 46)*

[Kohout, 2007] Kohout, J. (2007). On digital image representation by the delaunay triangulation. Dans *Proceedings of the 2nd Pacific-Rim Symposium on Image and Video Technology (PSIVT 07)*, pages 826–840. *(Cité page 32)*

[Kropatsch, 1994] Kropatsch, W. (1994). Building irregular pyramids by dual graph contraction. Dans *Vision, Image and Signal Processing*, pages 366–374. *(Cité page 30)*

[Kropatsch et Macho, 1995] Kropatsch, W. et Macho, H. (1995). Finding the structure of connected components using dual irregular pyramids. Dans *Discrete Geometry for Computer Imagery*, pages 147–158. *(Cité page 30)*

[Kukkonen et al., 1993] Kukkonen, H., Rovamo, J., Tiippana, K., et Näsänen, R. (1993). Michelson contrast, rms contrast and energy of various spatial stimuli at threshold. *Vision Research*, 33 : 1431–1436. *(Cité page 18)*

[Kwang Y. Lee, 2008] Kwang Y. Lee, M. A. E.-S., editor (2008). *Modern heuristic optimization techniques : theory and applications to power systems*, volume 39 de *IEEE Press Series on Power Engineering.* Wiley-IEEE. *(Cité page 38)*

[Leordeanu et Hebert, 2009] Leordeanu, M. et Hebert, M. (2009). Unsupervised learning for graph matching. Dans *Proceedings of the 2009 Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 09). (Cité page 78)*

[Levenshtein, 1966] Levenshtein, V. (1966). Binary codes capable of correcting deletions, insertions and reversals. *Soviet Physics Doklady*, 10 : 707. *(Cité pages 25 et 62)*

[Levi, 1972] Levi, G. (1972). A note on the derivation of the maximal common subgraphs of two directed or undirected graphs. *Calcolo*, 9 : 341–354. *(Cité page 80)*

[Lienhardt, 1991] Lienhardt, P. (1991). Topological models for boundary representation : a comparison with n-dimensional generalized maps. *Computer Aided Design*, 23 : 59–82. *(Cité pages 30 et 153)*

[Lienhardt, 1994] Lienhardt, P. (1994). N-dimensional generalized combinatorial maps and cellular quasi-manifolds. *International Journal on Computational Geometry and Applications*, 4(3) : 275–324. *(Cité page 153)*

[Lowe, 1999] Lowe, D. G. (1999). Object recognition from local scale-invariant features. Dans *Proceedings of the 7th International Conference on Computer Vision (ICCV 99)*, volume 2, pages 1150–1157. *(Cité page 21)*

[Lowe, 2004] Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60 : 91–110. *(Cité page 21)*

[Lozano et Escolano, 2006] Lozano, M. et Escolano, F. (2006). Protein classification by matching and clustering surface graphs. *Pattern Recognition*, 39(4) : 539–551. *(Cité page 57)*

[Lozano et Escolano, 2004] Lozano, M. A. et Escolano, F. (2004). Regularization kernels and softassign. Dans *Proceedings of the 9th Iberoamerican Congress On Pattern Recognition (CIARP 04)*, pages 320–327. *(Cité page 78)*

[Luo et al., 2003] Luo, B., Wilson, R., et Hancock, E. (2003). Spectral embedding of graphs. *Pattern Recognition*, 36(10) : 2213–2230. *(Cité pages 57 et 150)*

[Martin et al., 2001] Martin, D. R., Fowlkes, C., Tal, D., et Malik, J. (2001). A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. Dans *Proceedings of the 8th International Conference on Computer Vision (ICCV 01)*, volume 2, pages 416–423. *(Cité page 101)*

[McKay, 1981] McKay, B. D. (1981). Practical graph isomorphism. *Congressus Numerantium*, 30 : 45–87. *(Cité page 77)*

[Messmer et Bunke, 1999] Messmer, B. et Bunke, H. (1999). A decision tree approach to graph and subgraph isomorphism detection. *Pattern Recognition*, 32(12) : 1979–1998. *(Cité page 78)*

[Michelson, 1927] Michelson, A. A., editor (1927). *Studies in Optics*. University of Chicago Press. *(Cité page 18)*

[Mikolajczyk et Schmid, 2004] Mikolajczyk, K. et Schmid, C. (2004). Scale and affine invariant interest point detectors. *International Journal of Computer Vision*, 60(1) : 63–86. *(Cité page 21)*

[Mikolajczyk et al., 2005] Mikolajczyk, K., Tuytelaars, T., Schmid, C., Zisserman, A., Matas, J., Schaffalitzky, F., Kadir, T., et Gool, L. V. (2005). A comparison of affine region detectors. *International Journal of Computer Vision*, 65 : 2005. *(Cité page 21)*

[Milgram, 1967] Milgram, S. (1967). The small world problem. *Psychology Today*, 1 : 61. *(Cité page 7)*

[Moravec, 1977] Moravec, H. P. (1977). Towards automatic visual obstacle avoidance. Dans *Proceedings of the 5th International Joint Conference on Artificial Intelligence (IJCAI 77)*, page 584. *(Cité page 20)*

[Moulin et al., 2010] Moulin, C., Largeron, C., et Géry, M. (2010). Impact of visual information on text and content based image retrieval. Dans *Proceedings of the 13th IAPR International Workshop on Structural and Syntactic Pattern Recognition (SSPR 10)*, volume 6218 de *Lecture Notes in Computer Science*, pages 159–169. *(Cité page 9)*

[Munkres, 1957] Munkres, J. (1957). Algorithms for the assignment and transportation problems. *Journal of the Society for Industrial and Applied Mathematics*, 5(1) : 32–38. *(Cité page 83)*

[Myers et al., 2000] Myers, R., Wilson, R. C., et Hancock, E. R. (2000). Bayesian graph edit distance. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22 : 628–635. *(Cité page 83)*

[Nene et al., 1996] Nene, S. A., Nayar, S. K., et Murase, H. (1996). Columbia object image library (coil-100). Technical report, Columbia University CUCS-006-96. *(Cité page 52)*

[Neuhaus et Bunke, 2007] Neuhaus, M. et Bunke, H. (2007). Automatic learning of cost functions for graph edit distance. *Information Sciences*, 177(1) : 239–247. *(Cité page 83)*

[Neuhaus et al., 2006] Neuhaus, M., Riesen, K., et Bunke, H. (2006). Fast suboptimal algorithms for the computation of graph edit distance. Dans *Proceedings of the 11th International Workshop on Structural and Syntactic Pattern Recognition (SSPR 10)*, volume 4109 de *Lecture Notes in Computer Science*, pages 163–172. Springer. *(Cité page 83)*

[Nivre et McDonald, 2008] Nivre, J. et McDonald, R. T. (2008). Integrating graph-based and transition-based dependency parsers. Dans *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics : Human Language Technologies (ACL 08)*, pages 950–958. *(Cité page 7)*

[Nowak et al., 2006] Nowak, E., Jurie, F., et Triggs, B. (2006). Sampling strategies for bag-of-features image classification. Dans *Proceedings of the 9th European Conference on Computer Vision (ECCV 06)*, pages 490–503. Springer Berlin/Heidelberg. *(Cité page 23)*

[Pardo, 2002] Pardo, A. (2002). Semantic image segmentation using morphological tools. Dans *IEEE 2002 International Conference on Image Processing (ICIP 02)*, pages 745–748. *(Cité page 87)*

[Petrovic et al., 2002] Petrovic, S., Kendall, G., et Yang, Y. (2002). A tabu search approach for graph-structured case retrieval. Dans *Proceedings of STarting Artificial Intelligence Researchers Symposium (STAIRS 02)*, pages 55–64. *(Cité page 39)*

[Qiu et Sudirman, 2001] Qiu, G. et Sudirman, S. (2001). Color image coding, indexing and retrieval using binary space partitioning tree. Dans *IEEE 2001 International Conference on Image Processing (ICIP 01)*, pages 682–685. *(Cité page 27)*

[Riesen et Bunke, 2008] Riesen, K. et Bunke, H. (2008). Iam graph database repository for graph based pattern recognition and machine learning. Dans *Proceedings of the 12th IAPR International Workshop on Structural and Syntactic Pattern Recognition (SSPR 08)*, volume 5342 de *Lecture Notes In Computer Science*, pages 287–297. Springer-Verlag. *(Cité page 7)*

[Riesen et al., 2007] Riesen, K., Neuhaus, M., et Bunke, H. (2007). Bipartite graph matching for computing the edit distance of graphs. Dans *Proceedings of the 6th IAPR Workshop on Graph-based Representations in Pattern Recognition (GbRPR 07)*, volume 4538 de *Lecture Notes in Computer Science*, pages 1–12. *(Cité page 83)*

[Rissanen, 2007] Rissanen, J., editor (2007). *Information and complexity in statistical modeling*. Springer-Verlag. *(Cité page 86)*

[Robertson et al., 1997] Robertson, N., Sanders, D., Seymour, P., et Thomas, R. (1997). The four-colour theorem. *Journal of Combinatorial Theory, Series B*, 70 : 2–44. *(Cité page 7)*

[Ros et al., 2005] Ros, J., Laurent, C., Jolion, J.-M., et Simand, I. (2005). Comparing string representations and distances in a natural images classification task. Dans

Proceedings of the 5th IAPR Workshop on Graph-based Representations in Pattern Recognition (GbRPR 05), volume 3434 de *Lecture Notes in Computer Science*, pages 72–81. Springer Berlin/Heidelberg. *(Cité page 25)*

[Rosenfeld, 1974] Rosenfeld, A. (1974). Adjacency in digital pictures. Dans *Information and Control*, volume 26(1), pages 24–33. *(Cité page 29)*

[Rossignac, 1999] Rossignac, J. (1999). Edgebreaker : connectivity compression for triangle meshes. *IEEE Transactions on Visualization and Computer Graphics*, 5 : 47–61. *(Cité page 32)*

[Salembier et Garrido, 1998] Salembier, P. et Garrido, L. (1998). Binary partition tree as an efficient representation for filtering, segmentation and information retrieval. Dans *IEEE 1998 International Conference on Image Processing (ICIP 98)*, volume 2, pages 252–256. *(Cité page 27)*

[Samet, 1984] Samet, H. (1984). The quadtree and related hierarchical data structures. *ACM Computing Surveys*, 16(2) : 187–260. *(Cité page 26)*

[Samuel et al., 2010] Samuel, É., de La Higuera, C., et Janodet, J.-C. (2010). Extracting plane graphs from images. Dans *Proceedings of the 13th International Workshop on Structural and Syntactic Pattern Recognition (SSPR 10)*, volume 6218 de *Lecture Notes in Computer Science*, pages 233–243. *(Cité page 12)*

[Sanfeliu et Fu, 1983] Sanfeliu, A. et Fu, K.-S. (1983). A distance measure between attributed relational graphs for pattern recognition. *IEEE transactions on systems, man, and cybernetics*, 13(3) : 353–362. *(Cité page 82)*

[Schmid et al., 2000] Schmid, C., Mohr, R., et Bauckhage, C. (2000). Evaluation of interest point detectors. *International Journal of Computer Vision*, 37(2) : 151–172. *(Cité page 21)*

[Shapiro et Brady, 1992] Shapiro, L. S. et Brady, J. M. (1992). Feature-based correspondence : an eigenvector approach. *Image and Vision Computing*, 10(5) : 283–288. *(Cité page 78)*

[Sidibé et al., 2007] Sidibé, D., Montesinos, P., et Janaqi, S. (2007). Fast and robust image matching using contextual information and relaxation. Dans *International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications (VISAPP 07)*, pages 68–75. Institute for Systems and Technologies of Information, Control and Communication. *(Cité page 68)*

[Sidibe et al., 2008] Sidibe, D., Montesinos, P., et Janaqi, S. (2008). Matching local invariant features with contextual information : An experimental evaluation. *Electronic Letters on Computer Vision and Image Analysis*, 7(1) : 26–39. *(Cité page 68)*

[Singh et al., 2007] Singh, R., Xu, J., et Berger, B. (2007). Pairwise global alignment of protein interaction networks by matching neighborhood topology. Dans *Research in Computational Molecular Biology*, pages 16–31. *(Cité pages 53 et 78)*

[Smith et fu Chang, 1996] Smith, J. R. et fu Chang, S. (1996). Tools and techniques for color image retrieval. Dans *Proceedings of SPIE - Storage & Retrieval for Image and Video Databases*, volume 2670, pages 426–437. *(Cité page 24)*

[Smith et Brady, 1995] Smith, S. M. et Brady, J. M. (1995). Susan - a new approach to low level image processing. *International Journal of Computer Vision*, 23 : 45–78. *(Cité page 21)*

[Solnon, 2008] Solnon, C., editor (2008). *Optimisation par colonies de fourmis*. Programmation par contraintes. Hermès - Lavoisier. *(Cité page 39)*

[Solnon et Jolion, 2007] Solnon, C. et Jolion, J.-M. (2007). Generalized vs set median strings for histogram-based distances : algorithms and classification results in the image domain. Dans *Proceedings of the 6th IAPR Workshop on Graph-based Representations in Pattern Recognition (GbRPR 07)*, volume 4538 de *Lecture Notes in Computer Science*, pages 404–414. Springer Berlin/Heidelberg. *(Cité pages 25 et 40)*

[Sommellier, 1997] Sommellier, L. (1997). *Mise en correspondance d'images stéréoscopiques utilisant un modèle topologique*. Thèse de doctorat, Université Lyon I. *(Cité page 146)*

[Sorlin et Solnon, 2004] Sorlin, S. et Solnon, C. (2004). A global constraint for graph isomorphism problems. Dans *Proceedings of the 1st International Conference on Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems (CPAIOR 04)*, pages 287–302. *(Cité page 78)*

[Sorlin et Solnon, 2005] Sorlin, S. et Solnon, C. (2005). Reactive tabu search for measuring graph similarity. Dans *Proceedings of the 5th IAPR Workshop on Graph-based Representations in Pattern Recognition (GbRPR 05)*, volume 3434 de *Lecture Notes in Computer Science*, pages 172–182. Springer Berlin/Heidelberg. *(Cité page 39)*

[Stelldinger et al., 2006] Stelldinger, P., Köthe, U., et Meine, H. (2006). Topologically correct image segmentation using alpha shapes. Dans *Proceedings of the 13th International Conference on Discrete Geometry for Computer Imagery (DGCI 06)*, pages 542–554. *(Cité page 109)*

[Swain et Ballard, 1990] Swain, M. J. et Ballard, D. H. (1990). Indexing via color histograms. Dans *IEEE International Conference on Computer Vision (ICCV 90)*, pages 390–393. *(Cité page 24)*

[Swain et Ballard, 1991] Swain, M. J. et Ballard, D. H. (1991). Color indexing. *International Journal of Computer Vision*, 7 : 11–32. *(Cité page 23)*

[Ta, 2010] Ta, A. P. (2010). *Inexact graph matching techniques : Application to object detection and human action recognition*. Thèse de doctorat, Université de Lyon. *(Cité page 40)*

[Taubin et Rossignac, 1998] Taubin, G. et Rossignac, J. (1998). Geometric compression through topological surgery. *ACM Transactions on Graphics*, 17(2) : 84–115. *(Cité page 32)*

[Toussaint, 1980] Toussaint, G. T. (1980). The relative neighbourhood graph of a finite planar set. *Pattern Recognition*, 12 : 261–268. *(Cité page 31)*

[Tutte, 1963] Tutte, W. (1963). A census of planar maps. *Canadian Journal of Mathematics*, 15 : 249–271. *(Cité page 30)*

[Ullmann, 1976] Ullmann, J. R. (1976). An algorithm for subgraph isomorphism. *Journal of the ACM*, 23 : 31–42. *(Cité page 80)*

[Umeyama, 1988] Umeyama, S. (1988). An eigendecomposition approach to weighted graph matching problems. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 10(5) : 695–703. *(Cité pages 53 et 78)*

[Voronoi, 1907] Voronoi, G. (1907). Nouvelles applications des paramètres continus á la théorie des formes quadratiques. premier mémoire : sur quelques propriétés des formes quadratiques positives parfaites. *Journal für die Reine und Angewandte Mathematik*, 133 : 97–178. *(Cité page 165)*

[Voronoi, 1908] Voronoi, G. (1908). Nouvelles applications des paramètres continus á la théorie des formes quadratiques. deuxième mémoire : recherches sur les parallélloèdres primitifs. *Journal für die Reine und Angewandte Mathematik*, 134 : 198–287. *(Cité page 165)*

[Wallis et al., 2001] Wallis, W. D., Shoubridge, P., Kraetz, M., et Ray, D. (2001). Graph distances using graph union. *Pattern Recognition Letters*, 22 : 701–704. *(Cité page 81)*

[Wang et al., 2001] Wang, J. Z., Li, J., et Wiederhold, G. (2001). Simplicity : Semantics-sensitive integrated matching for picture libraries. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23 : 947–963. *(Cité page 53)*

[Wang et al., 2004] Wang, Z., Bovik, A. C., Sheikh, H. R., et Simoncelli, E. P. (2004). Image quality assessment : From error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4) : 600–612. *(Cité page 109)*

[Wilson, 1996] Wilson, R. J., editor (1996). *Introduction to graph theory.* Pearson Education. *(Cité page 70)*

[Zaslavskiy et al., 2008] Zaslavskiy, M., Bach, F., et Vert, J. (2008). A path following algorithm for the graph matching problem. Dans *Proceedings of 2008 International Conference on Image and Signal Processing (ICISP 08)*, pages 329–337. *(Cité pages 53 et 78)*

[Zeng et al., 2009] Zeng, Z., Tung, A. K. H., Wang, J., Feng, J., et Zhou, L. (2009). Comparing stars : On approximating graph edit distance. Dans *Proceedings of the 35th International Conference on Very Large Data Bases (VLDB 09)*, volume 2, pages 25–36. *(Cité page 83)*

Résumé

La reconnaissance de formes s'intéresse à la détection automatique de motifs dans des données d'entrée, afin de pouvoir, par exemple, les classer en catégories. La matière première de ces techniques est bien souvent l'image numérique. Cette dernière, dans sa forme la plus courante, est codée sous la forme d'une matrice de pixels.

Néanmoins, la question du développement de représentations plus riches se pose. Ainsi, la structuration de l'information contenue dans l'image devrait permettre la mise en évidence des différents objets représentés, et des liens les unissant.

C'est pourquoi nous proposons de modéliser les images numériques sous forme de graphes, pour leur richesse et expressivité d'une part, et pour exploiter les résultats de la théorie des graphes en reconnaissance de formes d'autre part. Nous développons pour cela une méthode d'extraction de graphes plans à partir d'images, basée sur le respect de la sémantique. Nous montrons que nous pouvons, étant donné un graphe, reconstruire avec perte limitée l'image d'origine.

Par la suite, nous introduisons les graphes plans à trous, graphes dont les faces peuvent être visibles ou invisibles. Leur justification trouve sa place dans la recherche de motifs notamment, pour laquelle les éléments constituant l'arrière plan d'une image ne doivent pas être retrouvés. En dirigeant notre attention sur la planarité de ces graphes, nous proposons des algorithmes polynomiaux d'isomorphisme de graphes plans et de motifs ; nous traitons également leur équivalence, qui se trouve être un isomorphisme aux faces invisibles près.

Abstract

Pattern recognition deals with automatically detecting patterns in input values, so as to, for example, classify them into categories. Digital images often consitute the raw material for these applications. The term *digital images* usually refers to bitmap images, *i.e.* images represented as matrices of pixels.

However, alternative representations can be considered. Thus, structuring the information contained in the image should underline the different objects depected in the image, as well as the links existing between them.

This is the reason why we propose to use graph-based representations. Indeed, on the one hand, graphs are complex data structures with important expressive power and, on the other hand, we should benefit from graphs theory results and apply them to pattern recognition tasks. To this extent, we develop a method for extracting semantically well-founded plane graphs from images. We show that it is possible to rebuild the original image from this kind of graphs, with limited loss.

Furthermore, we introduce open plane graphs, *i.e.* graphs whose faces can be visible or invisible. These graphs are useful in pattern recognition, when it is needed to search for patterns independently of the background. Focusing on the planarity of these graphs, we propose polynomial algorithms for plane graphs isomorphism and subgraphs isomorphism. We also address the equivalence issue, which is an isomorphism variant not taking into account visible faces.